COMBINATORIAL OPTIMIZATION ON MASSIVE DATASETS: STREAMING, DISTRIBUTED, AND MASSIVELY PARALLEL COMPUTATION

Sepehr Assadi

A DISSERTATION
in
Computer and Information Science
Presented to the Faculties of the University of Pennsylvania
in
Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy
2018

Supervisor of Dissertation

Sanjeev Khanna
Henry Salvatori Professor
Computer and Information Science

Graduate Group Chairperson

Rajeev Alur
Zisman Family Professor
Computer and Information Science

Dissertation Committee
Sampath Kannan (Chair), Henry Salvatori Professor, Computer and Information Science
Aaron Roth, Class of 1940 Bicentennial Term Associate Professor, Computer and Information Science
Val Tannen, Professor, Computer and Information Science
Michael Kapralov, Assistant Professor, School of Computer and Communication Sciences, Ecole Polytechnique Federale de Lausanne (EPFL)

COMBINATORIAL OPTIMIZATION ON MASSIVE DATASETS:
STREAMING, DISTRIBUTED, AND MASSIVELY PARALLEL COMPUTATION

Dedicated to my brother, Sepanta, and to Mina, the light of my life.

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my advisor Sanjeev Khanna for his perfect guidance, caring, engagement, patience, and support. Sanjeev and I spent countless hours meeting (literally) every day for the past few years on topics ranging from highly technical discussions on our joint projects or perhaps a cute problem one of us had recently encountered, to interesting research directions, and then to broader career and life advice. Sanjeev's utter commitment to my professional and personal development and countless hours of devotion to our work on one hand, and the freedom he gave me in my research on the other impacted me significantly. Being Sanjeev's student is the most cherished experience of my academic life and I am thankful every day for the privilege I had in having him as my advisor. I cannot thank you enough Sanjeev!

I am immensely grateful to Michael Kapralov, Val Tannen, and Omri Weinstein for additional mentoring over the last few years. Working with Michael and learning from him had a great impact on me and I had a lot of fun throughout our collaborations. Val has always been an endless source of advice, support, and encouragement since the early days of my academic life at Penn. Omri's support and never-ending commitment in general and his enthusiasm and great taste in choosing meaningful problems to work on in particular made collaborating with him one of my most rewarding experiences.

I want to further express my gratitude to Sampath Kannan, Michael Kapralov, Aaron Roth, and Val Tannen for devoting their time and serving on my thesis committee, and for their frequent advice and suggestions. I am also indebted to CS department staff at Penn, specially Mike Felker, Cheryl Hickey, and Lily Hoot, for all their help.

I am deeply thankful to my collaborators over the past few years: To Arpit Agarwal, Shivani Agarwal, Eric Balkanski, MohammadHossein Bateni, Aaron Bernstein, Deeparnab Chakrabarty, Yu Chen, Julia Chuzhoy, Justin Hsu, Shahin Jabbari, Sampath Kannan, Michael Kapralov, Sanjeev Khanna, Yang Li, Vahab Mirrokni, Krzysztof Onak, Renato Paes Leme, Victor Preciado, Baruch Schieber, Shay Solomon, Cliff Stein, Xiaorui Sun, Val Tannen, Rakesh Vohra, Omri Weinstein, Grigory Yaroslavtsev and Qin Zhang. I am particularly thankful to Yang Li for our collaborations at the start of my research in graduate school and for being truly an "older academic sibling" to me. I also like to thank Vova Braverman, Piotr Indyk, Ran Raz, David Woodruff, and Morteza Zadimoghaddam for many illuminating discussions and advice. Furthermore, I am truly grateful to Sepideh Mahabadi and Ali Vakilian for their help and advice on many occasions starting from my undergraduate years.

I would like to thank MohammadHossein Bateni and Vahab Mirrokni for being my mentors during a summer internship at Google Research, and Krzysztof Onak, Baruch Schieber, and Shay Solomon for hosting me during a short but fruitful visit to IBM research.

I further thank Mohammad Ghodsi, Mohammad Izadi, and Hamid Zarrabi-Zadeh for their help and support at the beginning of my academic life as an undergraduate at Sharif University and for forming my interest in theoretical computer science.

A big thank you to my friends in the last five years at Penn: Nimit, Shahin, Alyssa, Mahyar, Hoda, Rachel, Jamie, Steven, and Justin. Special thanks to Nimit for being my friend right from the start of my new life away from home, to Alyssa for countless hours of wandering around in Philly, and to Shahin for being super helpful with everything and introducing me to indoor rock climbing. Many thanks also to Eric, Manolis, Naman, Sadra, Thodoris, Zelda, and other interns at Google for their invaluable friendship. I also like to thank a few of my friends over the years that certainly made their marks on my life: to Ashkan, Ehsan, Nooshin, and Saeid for being the greatest friends one could ever hope for and for always being there for me even though life had kept us mostly apart.

Lastly, a huge thanks to my dear parents Aman and Vida, and to my younger brother and best friend Sepanta, for their consistent encouragement, support, and understanding throughout all my life. My only complaint about the last few years is that I did not have you as much as I wanted in my life. I am so sorry that I could not visit you during this time. Last but not the least, I am beyond words of gratitude for the two constants of my life, to my furry four-pawed princess Shasta for her unconditional love and much needed comic breaks, and to Mina, for being the better half of me, and for her unbelievable amount of love, encouragement and support.

ABSTRACT

COMBINATORIAL OPTIMIZATION ON MASSIVE DATASETS: STREAMING, DISTRIBUTED, AND MASSIVELY PARALLEL COMPUTATION

Sepehr Assadi

Sanjeev Khanna

With the emergence of massive datasets across different application domains, there is a rapidly growing need to solve various optimization tasks over such datasets. This raises the following fundamental question: *How well can we solve a large-scale optimization problem on massive datasets in a resource-efficient manner?* The focus of this thesis is on answering this question for various problems in different computational models for processing massive datasets, in particular, streaming, distributed, and massively parallel computation models.

The first part of this thesis is focused on **graph optimization**, in which we study several fundamental graph optimization problems including matching, vertex cover, and connectivity. The main results in this part include a tight bound on the space complexity of the matching problem in dynamic streams, a unifying framework for achieving algorithms that improve the state-of-the-art for matching and vertex cover problems in all the mentioned models, and the first massively parallel algorithm for connectivity on sparse graphs that improve upon the classical parallel PRAM algorithms for a large family of graphs.

In the second part of the thesis, we consider **submodular optimization** and in particular two canonical examples of set cover and maximum coverage problems. We establish tight bounds on the space complexity of approximating set cover and maximum coverage in both single- and multi-pass streams, and build on these results to address the general problem of submodular maximization across all aforementioned models.

In the last part, we consider **resource constrained optimization** settings which require computation over data which is not particularly large but still imposes restrictions of similar nature. Examples include optimization over data which can only be accessed with limited adaptivity (e.g. in crowdsourcing applications), or corresponds to private information of agents and is not available in an integrated form (e.g., in auction and mechanism design applications). We use the toolkit developed in the first two parts of the thesis to obtain several new results for such problems, significantly improving the state-of-the-art.

A common theme in this thesis is a rigorous study of problems from both an algorithmic perspective and impossibility results. Rather than coming up with ad hoc solutions to problems at hand, the goal here is to develop general techniques for solving various large-scale optimization problems in a unified way in different models, which in turns requires further understanding of the limitations of known approaches for addressing these problems.

List of Tables

List of Illustrations

Chapter 1
Introduction

Massive datasets abound. For instance, Google has been processing more than 20 Petabytes of data on a day-to-day basis for nearly a decade now [115], Facebook graph has contained more than trillion edges for quite some time [100], and in some applications, we now need to process streams of data at a rate of Gigabytes per second [91]. The need to process such massive datasets necessitates rethinking the task of algorithm design as the traditional computational models do not capture these scenarios accurately. For instance, it is no longer realistic to assume the standard *random access model* where the data can be accessed in an arbitrary order and that each data access has unit cost. As a result, we are now encountering a paradigm shift toward modern computational models that can more accurately capture the essence of computation on massive datasets.

A salient challenge in processing massive datasets is that the entire input is orders of magnitude larger than the amount of storage on one processor. To address this challenge, several different computational models have been introduced, each focusing on certain additional *resources* needed to solve large-scale problems. Examples include the streaming model, the distributed communication model, and the massively parallel computation (MPC) model that is a common abstraction of MapReduce-style computation. The main target resources in these models are the (internal) *memory* in case of streaming algorithms, the *communication* in distributed algorithms, and both memory and communication, as well as number of *rounds* of computation, for MPC algorithms.

This thesis we will be concerned with the theoretical challenges in solving *combinatorial optimization* problems on massive datasets. Combinatorial optimization appear in an amazingly vast variety of domains including in computer science, operations research, data science, economics, and numerous others. The emergence of massive datasets in recent years has changed the nature of the optimization tasks that we are facing. Classical algorithms for well-studied optimization problems, say maximum matching or set cover, no longer result in efficient solutions in modern computational models. At the same time, previous notions of "hardness" of a problem, e.g., NP-hardness, are also not necessarily a good indicator of the complexity of a problem in these modern models, which focus primarily on resources other than computation time. It is therefore critical to study these problems with an eye towards modern computational models: how to design new and provably efficient algorithms for different problems across these modern computational models? What are the barriers to efficiency or inherent limitations to optimization in these models? What are some general techniques applicable to a wide variety of large-scale optimization problems?

The research presented in this thesis stems from pursuing the above questions for two general family of combinatorial optimization problems, namely *graph optimization* and *submodular optimization*.

Massive graphs are ubiquitous. For example, both the web graph and models of the human brain use around 10^{10} nodes [242]. Analyzing massive graphs via classical algorithms can be challenging given the sheer size of these graphs. As such, there has been extensive interest in studying graph optimization problems in different models of computation over massive datasets. Submodular optimization also encompasses a wide variety of problems in combinatorial optimization including set cover, maximum coverage, minimum/maximum cut, to name a few. These problems have various applications in different areas including in machine learning, operations research, information retrieval, data mining, and web host analysis. As a result, submodular optimization has also received quite a lot of attention in models of computation over massive datasets.

We describe new techniques for developing algorithms and impossibility results for graph optimization and submodular optimization problems on massive datasets, with the following broad consequences:

- a general algorithmic approach for graph optimization applicable to all three models discussed above that bypasses the impossibility results known for prior techniques;
- the first MPC algorithm for a graph problem that breaks the well-known linear-memory barrier in this model;
- the first non-trivial impossibility result for a graph or submodular optimization problem in dynamic streams that can be solved "easily" in insertion-only streams;
- a new framework for proving impossibility results for multi-round distributed algorithms and multi-pass dynamic streaming algorithms.

Using these techniques we analyze a variety of central optimization problems in modern computational models for processing massive datasets and closely related models, and obtain improved algorithms and impossibility results for the following:

- the maximum matching problem;
- the minimum vertex cover problem;
- the graph connectivity problem in undirected graphs;
- the problem of estimating rank of a given matrix;
- the set cover problem;
- the maximum coverage problem;

- constrained submodular maximization;
- welfare maximization in combinatorial auctions;
- best arm identification in stochastic multi-armed bandits;
- ranking from pairwise comparisons;

A common theme in this thesis is a rigorous theoretical study of the problems from both an algorithmic perspective and impossibility results. Rather than coming up with an ad hoc solution to each problem at hand, the goal here is to develop general techniques for solving large-scale optimization problems in a unified way in different computational models. This can only be achieved by understanding the powers and limitations of current algorithmic approaches for solving these problems to know when and why these techniques fail and how a new approach can circumvent such limitations.

In the rest of this section, we elaborate more on our results and backgrounds. We start by formally defining the modern computational models that we work with in this thesis in Section 1.1. In Section 1.2, we review some of the research that has been done in this area in the past. Section 1.3 next describes our main contributions in this thesis. A short outline of the remainder of this thesis appears in Section 1.4.

1.1. Computational Models for Processing Massive Datasets

The three main computational models that we consider in this thesis are *streaming*, *distributed communication*, and *massively parallel computation (MPC)* model. We define these models below and review some basic backgrounds about each. We then describe some of the main connections between these models and general algorithmic approaches that are applicable to all these models simultaneously.

1.1.1. The Streaming Model

Data streams are ubiquitous. Examples include the network traffic flowing past a router, files read from an external memory device, or data transmitted by a satellite. The streaming model abstracts the main algorithmic constraint when processing such data: sequential access to data and limited working memory.

Let us start by an example. Consider a collection of large unstructured data files such as search-engine log-files or biological data. As these files are too large to fit the main memory of a computer, they need to be stored on external devices such as massive hard-drives. The problem that arises is that access to data on such devices can be very slow if the data is accessed in an arbitrary order, the common assumption of random order access in classical algorithms. This is because while such devices have reasonable transfer rates, i.e., data can be transferred sequentially at speed, the seek times of these devices is often prohibitively

large. In this setting, having a streaming algorithm that is able to process the data in these files as a stream of sequential accesses, while using a limited working memory (that resides on the main memory of a computer) can lead to a dramatic boost in the efficiency.

We now define the streaming model formally. For our purpose, we simply consider a data stream as a long sequence of data items, $A = \langle a_1, \ldots, a_m \rangle$ where each data item a_i comes from some large universe $\mathcal{U}$. Depending on the application, this universe $\mathcal{U}$ can be, say, numerical values in some range, a collection of subsets of a known ground-set, or edges of a graph, to mention a few. The goal is to compute a known function of the data stream, which in the context of the examples above can be, number of distinct elements in A, a minimum set cover of the ground-set using the sets in A, or a maximum matching in the graph defined by A. In order to do this, the algorithms are allowed to make (preferably) one or a few passes over the stream A, while using a *limited* memory which is (much) smaller than both A and $\mathcal{U}$. The algorithm outputs the solution after the last pass over the stream.

As is clear from the discussion above, the main resources of interest in streaming algorithms are ***number of passes*** over the input and ***memory requirement or space complexity*** of the algorithm. In different settings the limits on these resources vary. For instance, in external memory applications, multiple passes over the input maybe feasible, whereas processing the traffic flow of a router inherently requires a single-pass streaming algorithm; additionally, in the former example, a working memory that is, say, quadratically smaller than the input size may be accommodated while in the latter application one typically needs algorithms with space complexity exponentially smaller than the input size. We remark that for typical optimization problems in the streaming model—the focus of this thesis—the former setting of parameters, i.e., allowing multiple (yet small) number of passes over the input and memory requirement which is smaller only by a polynomial factor from the input size is the main setting of interest.

Stream order. One important facet of the streaming model is how the items are ordered in the data stream. A common approach here is to make no assumption about the ordering of the items arriving in data stream, a setting often referred to as *adversarially-ordered* streams. However, this approach is often too pessimistic and may limit the power of algorithms to the point of ruling out any non-trivial theoretical guarantees. As such, depending on the underlying application, it is sometimes preferable to consider *random order* streams in which the ordering of the m items in data stream is chosen uniformly at random from all $m!$ possible orderings.

Dynamic stream. The definition of data stream provided above is referred to as *insertion-only* streams as the data stream is only inserting new items to the underlying instance of the problem. Another widely studied model is *dynamic streams* (also referred to as *turnstile*

streams) in which both insertion and deletion of items are allowed in the stream, and the goal is to solve the problem on the final set of data items present. More formally, each entry of the data stream is a pair (a, Δ) for $a \in \mathcal{U}$ and $\Delta \in \{-, +\}$, and the interpretation is that the item a is inserted to input whenever $\Delta = +$ and is deleted otherwise. We assume that no item is ever deleted in the stream before being inserted (a model known as *strict* turnstile model). It is immediate to verify that dynamic streams is a generalization of insertion-only streams. We study both models in this thesis.

The streaming model evolved primarily over the course of three papers [17, 179, 141] although it has its root in several older papers [251, 253, 145]. We refer the interested reader to a survey by Muthukrishnan [254] for more details.

1.1.2. The Distributed Communication Model

Oftentimes a massive dataset is distributed across multiple machines, which are interconnected by a communication network, and to process the data, the machines need jointly compute a function defined on the union of their inputs by exchanging messages with each other. For instance, social network graphs are usually stored on many different machines and to answer queries such as whether the whole graph is connected, we have to synthesize data from all the machines. The distributed communication model is a common abstraction of computation in these scenarios, focusing primarily on the amount of *communication* between machines, and the number of *rounds* of communication (under various constraints on what messages can be sent by each machine in each round).

We now define this model formally. Let $A = (a_1, \ldots, a_m)$ be a collection of input data items chosen from a universe $\mathcal{U}$. As before, these items can be anything ranging from numerical values to edges of some graph. The input A is partitioned across k machines $P^{(1)}, \ldots, P^{(k)}$ sometimes referred to as players. We assume additionally that there exists an additional party, called the *coordinator*, who receives no input. Players can only communicate with the coordinator and not with each other directly. The addition of coordinator is only for simplifying the model: instead of player $P^{(i)}$ sending a message x to player $P^{(j)}$ directly, $P^{(i)}$ can simply send the message (x, j) to the coordinator and the coordinator can relay this message to $P^{(j)}$[1]. Communication happens over synchronous rounds. In each round, all players simultaneously send a message to a central coordinator who then communicates back to all machines to guide the computation for the next round. We sometimes refer to distributed algorithms as *protocols* also. We make the standard assumption that the players in this model have access to *public randomness*, i.e., a *shared* tape of randomness bits that they can all access.

[1]Note that this "simulation" increases the communication by a multiplicative factor of $O(\log k)$ which is typically negligible.

Similar in spirit to the assumptions made on the ordering of data streams in the streaming model, here also one can make different assumptions about the *partitioning* of the input across different machines. Two common assumptions are *adversarial partitions*, where data items can be partitioned arbitrarily, and *random partitions*, where each item is sent to one of the machines chosen uniformly at random.

The main computational resources of interest in the distributed communication model are ***round complexity*** and ***communication complexity***, which are respectively, the total number of communication rounds, and the total communication by players measured in the number of bits communicated. It is easy to see that communication *linear* in input size always allows for solving the problem in one round of communication. Alas, such a communication is prohibitively expensive in case of massive inputs—the focus of this thesis—and hence in this model we are interested in protocols with (much) smaller communication complexity than the input size.

We note that, depending on the application, different variations of the distributed communication model we defined above have been studied in the literature (the model that we adopt is typically referred to as the *coordinator* model in the literature, see, e.g. [268]). The choice of this particular model in this thesis is tailored specifically towards large-scale optimization on massive datasets to capture the necessity of obtaining protocols with low-cost communication complexity that are "fast" i.e., finish the computation in a small number of *parallel* rounds of communication.

1.1.3. The Massively Parallel Computation Model

For over a decade now, we have witnessed the emergence of various parallel computing platforms such as MapReduce, Hadoop and Spark for processing massive datasets (see, e.g. [115, 299, 309]). A key differentiating feature of these platforms from previous (theoretical) models of parallel computation is that they interleave sequential and parallel computation by allowing much more local computational power and storage to processors compared to classical settings such as PRAM. The massively parallel computation (MPC) model has been proposed in the theory community to capture this key feature of these platforms to allow an algorithmic approach to these platforms.

Let $A = (a_1, \ldots, a_m)$ again be a collection of input data items chosen from some known universe $\mathcal{U}$. In the MPC model, we have p machines each with local memory s. The input is originally partitioned across the machines in a load balanced way. It is required that both the number of machines and the local memory of each machine to be at most $m^{1-\delta}$ for some constant $\delta > 0$, and that the total memory of the system be at most $\widetilde{O}(m)$, i.e., proportional (within logarithmic factors) to the total input size[2]. The motivation behind

[2]We implicitly assumed that each data item from universe $\mathcal{U}$ can be represented by polylog(m) bits and

these constraints is that the number of machines, and local memory of each machine should be much smaller than the input size to the problem since these frameworks are used to process massive datasets.

Computation in this model proceeds in synchronous rounds. During a round, each machine runs a local algorithm on the data assigned to it. No communication between machines is allowed during a round. Between rounds, machines are allowed to communicate so long as each machine send or receive a communication no more than its memory. Any data output from a machine must be computed locally from the data residing on the machine (this data includes the previous messages received and stored by the machine as well). The primary resource of interest in the MPC model is the total number of rounds, referred to as the ***round complexity*** of the algorithm.

The MPC model was initially introduced by Karloff *et al.* [213] and was further refined in a series of work [161, 23, 53]. In this thesis, we adopt the most stringent (as well as the most popular) definition of this model due to Beame *et al.* [53].

We point out that the MPC model is a special case of the Bulk-Synchronous-Parallel (BSP) model [294], but has the advantage of having fewer parameters. This makes algorithm design more "coarse-grained" and streamlines the search for efficient algorithms, as evident by the omnipresence of this model in practice.

Comparison with PRAM algorithms. Before we move on from this section, it is imperative to compare the power of MPC algorithms with the classical parallel algorithms in the PRAM model. It was first shown in [213] that one can simulate T steps of an EREW PRAM algorithm by an MPC algorithm in $O(T)$ MPC rounds. Subsequently, it was show in [161] how to simulate T steps of CRCW PRAM algorithms (the most general family of PRAM) in $O(T \cdot \log_s M)$ MPC rounds where s is the per-machine memory of the MPC algorithm and M is the total memory used by the PRAM algorithm. Finally, in [277], it is shown that even "weak" lower bounds on the round complexity of MPC algorithms would imply "strong" lower bounds in the PRAM model. We defer the exact technical details of this result to later chapters but mention that this result implies that any $\omega(1)$ round lower bound in the MPC model when machines have memory which is only polynomially smaller than the input size would imply that $\mathbf{NC}^1 \subsetneq \mathbf{P}$, a major breakthrough in complexity theory which seems way beyond the scope of current techniques.

1.1.4. Connections Between These Models

The three models above, along with their seemingly different target resources, turn out to be closely related. Below, we describe some of the obvious and not-so-obvious connections

hence the input size is $\widetilde{O}(m)$ indeed. Throughout this thesis, this is always going to be the case. However, in general, one should use the input size in the discussion above instead of the parameter m.

between these models. We start by reviewing some general algorithmic techniques that target all these models simultaneously and then present some results that extend impossibility results in one model to another ones.

General Techniques for One Round/Pass Algorithms

A basic and abstract algorithmic approach to large-scale optimization is as follows: partition the data into multiple pieces, compute a representation or a summary of each piece, merge the summaries together and recover the solution from the merge without further accessing the original input. The main (and the most non-trivial) step in this approach is the second one that constructs representations that are on one hand "small" and on the other hand "mergeable" and allow for recovering an approximate solution to the original problem. One can implement this idea in each of the models above to obtain efficient algorithms:

- Distributed communication model: The data is already partitioned into multiple pieces in this model. Each machine can thus compute a small and mergeable representation of its input and send it to the coordinator. The coordinator can then merge these summaries and recover the solution from them. It is immediate that the communication complexity of this protocol is proportional to size of the representations and that this protocol is *round optimal*, i.e., only requires one round of communication.
- MPC model: The idea here is quite the same as above. By partitioning the data into multiple pieces, we can ensure that each piece would fit the small memory of each machine, and by ensuring the size of representations are small, we can fit *all* the representations on a single machine and solve the problem.
- Streaming model: The application to streaming is slightly more subtle but still quite simple. This time, we partition consecutive parts of an insertion-only stream into multiple pieces. Next, we read one piece of the stream at a time and store it entirely in the memory and then compute its representation. After that we store this representation in the memory and discard the stored piece of stream and read the next one. As a result, the memory requirement of this algorithm is proportional to size of *one* piece of the stream, and the total size of small representations.

We point out that a key feature of this approach is that it can be implemented with a *minimal* number of rounds or passes, i.e., only *one*, in the aforementioned models.

Two particularly successful techniques for designing small and mergeable representations in above strategy are *linear sketches* and *composable coresets*. Linear sketching technique corresponds to taking a linear projection of the input data as its representative summary. The "linearity" of the sketches is then used to obtain a sketch of the combined pieces from which the final solution can be extracted. Coresets on the other hand are sub-

sets of the input that suitably preserve properties of the input data, and they are said to be composable if the union of coresets for each input piece yields a coreset for the union of the pieces. We now define each technique in more detail.

Linear sketches. Let $A = (a_1, \ldots, a_m)$ be a collection of items from a universe $\mathcal{U}$ with size n. The *frequency vector* of A is an n-dimensional vector $f_A \in \mathbb{N}^n$ where the i-th entry denotes the number of times the i-th item of $\mathcal{U}$ appears in A. Let $M \in \mathbb{R}^{d \times n}$ be a matrix (possibly chosen randomly). Then, the d-dimensional matrix $M \cdot f_A$ is called a *linear sketch* of f_A (or A). The goal is to design a distribution over matrices M such that the solution to a particular problem on A, say the number of distinct elements in A, can be recovered from the linear sketch $M \cdot f_A$ with no further access to A. It is easy to see that by setting M to be the $n \times n$ identity matrix this task can be done trivially. However, linear sketches are interesting when the parameter d, i.e., the number of rows of the sketching matrix M is much smaller than n, and hence the resulting sketch is much smaller than f_A (throughout, we always assume that the bit-representation of entries of M are $O(\log n)$).

The approach explained earlier in this section can be used to obtain efficient distributed communication, MPC algorithms, and streaming algorithms using linear sketches. However, one can in fact use linear sketches in a more clever way to obtain even more efficient streaming algorithms in the most general variant of the model, i.e., dynamic streams. The algorithm picks a sketching matrix M at the beginning of the stream and upon any incoming update (a_i, Δ_i) of the dynamic stream, the linear sketch is updated to $M \cdot f_i = M \cdot f_{i-1} + \Delta \cdot M \cdot \mathbf{1}_a$ where f_i is the frequency vector of the underlying input at this point, f_{i-1} is the one before receiving the i-th update, and $\mathbf{1}_a$ is an n-dimensional vector which is only one for the entry corresponding to item $a \in \mathcal{U}$. This way, the algorithm can maintain a linear sketch of the underlying input and at the end of the stream use it to solve the problem. The memory requirement of this algorithm is $O(d)$ to store the linear sketch plus the space needed to (implicitly) store the matrix M—typically much smaller by a careful careful choice of the sketching matrix with limited independence).

We refer the interested reader to surveys by McGregor [242] and Woodruff [301] for further applications of linear sketches and more details.

Composable coresets. Composable coresets were first introduced by Indyk *et al.* [186]. Let $\mathcal{U}$ be a universe of n items. We say that an algorithm ALG outputs a *composable coreset* for some fixed optimization problem, if given any $A \subseteq \mathcal{U}$, ALG outputs a subset $\mathsf{ALG}(A) \subseteq A$ in a way that for any collection $A_1, \ldots, A_k$ of subsets of $\mathcal{U}$, the optimal solution on $\mathsf{ALG}(A_1) \cup \ldots \cup \mathsf{ALG}(A_k)$ is (approximately) equal to the optimal solution on $A_1 \cup \ldots \cup A_k$. Intuitively speaking, this means that the $\mathsf{ALG}(A)$ preserves the optimal solution in A, i.e., is a "coreset", and that $\mathsf{ALG}(A) \cup \mathsf{ALG}(B)$ is as (almost) as good as

$\mathsf{ALG}(A \cup B)$, i.e., is "composable". A trivial composable coreset would be to store A as its coreset, however, similar to linear sketches (and all other representations), the interesting case is when the size of coreset is much smaller than the size of the original input.

Let us give an example of a composable coreset in the context of graph optimization problems for the minimum spanning tree problem. We define ALG as an algorithm that given any graph $G(V, E)$, outputs an arbitrarily minimum spanning tree of G as $\mathsf{ALG}(G)$. We argue that ALG indeed outputs a composable coreset for the minimum spanning tree problem that allows for recovering an exact optimum solution. Consider $\mathsf{ALG}(G_1) \cup \ldots \cup \mathsf{ALG}(G_k)$ for any collection of k graphs $G_1, \ldots, G_k$ on the same set of vertices and let $G := G_1 \cup \ldots \cup G_k$. A minimum spanning tree of G can be found using only the edges in $\mathsf{ALG}(G_1) \cup \ldots \cup \mathsf{ALG}(G_k)$ by picking any arbitrary minimum spanning tree of this subgraph of G (this can be proven easily by, for example, by considering the execution of the Kruskal's algorithm on G and noticing that it never needs any edge outside the union of the coresets). We point out that various other fundamental graph problems such as connectivity, sparsifiers, and spanners admit similar natural composable coresets.

We note that in addition to linear sketches and composable coresets, techniques such as *sampling* or *mergable summaries* introduced by Agarwal *et al.* [5] are also related to the approaches discussed in this section, but are less applicable to optimization problems.

General Techniques for Multi-Round/Pass Algorithms

One can extend the general approach in the previous part to multi-round/pass algorithms in all the three models discussed in this thesis. The high-level idea is, instead of recovering a solution directly from the merged representations, use the representations gathered in the first round/pass to guide our choice of the next set of representations. For example, in the application to the distributed communication model, after receiving the representations from all the machines, the coordinator sends back a summary of these messages to every machine so as to guide the construction of representations for the next round.

Sample-and-prune. A more concrete instantiation of this idea is the *sample-and-prune* technique of Kumar *et al.* [223] for implementing *greedy* algorithms in these three models and its special case for graph problems, referred to as *filtering*, by Lattanzi *et al.* [225]. We describe this technique in the distributed communication model, but extending this idea to both MPC and streaming models is straightforward.

Consider a greedy algorithm for some problem that picks a set of items sequentially and irrevocably in the solution (think of the greedy algorithm for finding a maximal matching or a minimum spanning tree of a graph). The idea is to first sample a relatively large subset of the items in the input across the machines and send them to the coordinator. The

coordinator then run the greedy algorithm on these items to pick a partial solution and this partial solution is shared with the machines. In the round, the machines discard all items that no longer can be picked by the greedy algorithm given this partial solution and repeat the sampling process again, until the coordinator is able to append its partial solutions obtained over rounds to a complete solution. We refer the interested reader to [223] for further details and extension of this idea to streaming and MPC models.

Adaptive linear sketching. Another widely used technique following this general approach is *adaptive* linear sketching. As in the previous part, we describe this technique in the distributed communication model while noting that extending it to MPC or streaming algorithms is straightforward.

In this technique, the machines first jointly pick a sketching matrix M using public randomness, and each machine $P^{(i)}$, sends the linear sketch $M \cdot f_{A_i}$ to the coordinator, where f_{A_i} is the frequency vector of the input set A_i on this machine. This way, the coordinator can compute $M \cdot f_A$ for $A := A_1 \cup \ldots \cup A_k$, i.e., the linear sketch for the whole input set (by linearity of the sketches). This sketch is then shared with all machines. The machines then pick another sketching matrix adaptively from the distribution of matrices conditioned on the first linear sketch. This process is repeated over multiple rounds until the coordinator outputs the final solution. We refer the interested to [10] and [209] for some instantiations of this technique and further details.

Connections Between Impossibility Results Across Models

One can establish many simple (but perhaps not so useful) connections between these three models. For instance, it is easy to see that any streaming algorithm with s bits of memory leads to a distributed algorithm with $O(s \cdot p)$ communication per machine (p is the number of passes of the streaming algorithm) and the total number of rounds equal to p times the number of machines. We leave establishing similar-in-spirit connections between streaming and MPC algorithms or between distributed protocols and MPC algorithms as an exercise.

However, there are also many more intricate connections between these models. Most related to the topics in this thesis, are two amazing results by Li *et al.* [230] and Ai *et al.* [14] that present a characterization of dynamic streaming algorithms in terms of linear sketches and prove that any dynamic streaming algorithm with space s can be transformed into a distributed protocol with $O(s \cdot p)$ communication per machine (p is the number of passes of the streaming algorithm) and *the same exact* number of rounds as the number of passes of the dynamic streaming algorithm. The transformation of the streaming algorithm is quite non-trivial and computationally expensive and so this result may not be considered as a useful algorithmic approach for transferring the algorithms from dynamic streaming model to distributed setting. However, an extremely useful implication of this result is that

impossibility results in the distributed communication model also imply impossibility results in the dynamic streaming model. This is a highly non-trivial fact and can be exploited to prove impossibility results in the dynamic streaming model which are not achievable by previous techniques. We use this direction of these results in multiple places in this thesis. Finally, by the remark made at the end of Section 1.1.3, achieving similar connection for establishing impossibility results in the MPC model seems quite unlikely.

1.2. Overview of the Previous Research

Designing algorithms for optimization problems on modern computational models for processing massive datasets has been a highly active area of research for more than a decade now. In what follows, we give a brief overview of the research that has been done in this area. In subsequent sections and chapters, we shall go into further detail about some of the work that is directly relevant to the results presented in this thesis. We emphasize that what follows is by no means a comprehensive list of all results in this area. We further point out due to different connections between these models discussed in Section 1.1.4, many of the results in one model can be translated to the other models as well and hence we may not explicitly repeat these results again for each model.

1.2.1. Streaming Algorithms

Traditionally, the main focus in the streaming model was on *numerical estimation* problems such as frequency moment estimations, quantiles, heavy hitters, and many others, and this still remains an active area of research in this model (we refer the reader to [254] for a survey of initial results). However, in the last decade or so, extensive attention in the streaming model have been dedicated to optimization problems as well.

Graph problems have been considered in the streaming model starting from one of the earliest papers in this model by Henzinger *et al.* [179] (see also [45]). This trend got momentum by a paper of Feigenbaum *et al.* [139] that identified a "sweet-spot" for streaming algorithms for processing graphs, the so-called *semi-streaming* algorithms. These are algorithms that on an n-vertex graph are allowed to use $n \cdot \text{polylog}(n)$ memory, which is quadratically smaller than the input size when the graph is dense, i.e., has $\Omega(n^2)$ edges. Feigenbaum *et al.* [139] showed that most graph problems including connectivity, minimum spanning tree, matching, etc., require $\Omega(n)$ space and that some of these problems can also be also solved or approximated in $n \cdot \text{polylog}(n)$ memory, hence justifying this choice of parameters. Since then, numerous graph problems have been studied in the streaming model including connectivity and minimum spanning tree [139], shortest path and diameter [140, 170], maximum matchings [139, 241, 125, 221, 160, 204, 267, 198], minimum and maximum cuts [310, 206, 207], sparsification [7, 217], spanners [127, 48, 126], maximum independent sets and cliques [171, 172], subgraph counting [45, 82, 202, 77, 246], random walks [282], and

many more. We refer the reader to these papers and references therein for more information.

The results mentioned above all targeted *insertion-only* streams. The next wave of results started with two papers by Ahn, Guha, and McGregor [11, 12] that introduced for the first time the idea of *graph sketching*, i.e., linear sketches for graph problems. This resulted in a flurry of results extending previous algorithms in insertion-only streams to dynamic graph streams (recall that linear sketches immediately imply efficient streaming algorithms for dynamic streams); see, e.g. [11, 12, 13, 209, 208, 102, 61, 169, 243, 182] and references therein for dynamic streaming algorithms for a wide range of problems such as connectivity, minimum spanning tree, sparsification, spanners, subgraph counting, etc.

Different problems in the family of submodular optimization have also been studied extensively in the streaming model. For instance, Saha and Getoor initiated the study of the set cover problem in the streaming model [281] which evolved into an active area of research [281, 108, 128, 116, 187, 174, 90, 50]. Another widely studied set of problems is submodular maximization under various constraints such as cardinality, matroids, p-systems, etc; see, e.g. [223, 87, 40, 93, 129, 247] and references therein.

1.2.2. Distributed Communication Algorithms

The distributed communication model presented in Section 1.1.2 has been studied extensively in recent years (see, e.g., [268, 67, 302, 303, 304, 183], and references therein). Traditionally, the focus in this model has been on optimizing the communication cost and round complexity issues have been typically ignored. However, in recent years, motivated by application to big data analysis, there have been growing interest in studying round efficient protocols in this model as well (see, e.g., [11, 12, 209, 186, 165, 249, 112, 166]).

Both graph optimization and submodular optimization problems have been studied considerably in the distributed communication model. For instance, it was shown in [268] and [183] that solving connectivity problem or even a weak polylog(n)-approximation to matching, respectively, requires $\Omega(nk/\text{polylog}(n))$ communication regardless of number of rounds (n is the number of vertices in the graph and k is the number of machines). Another example is the results of [186, 249, 112] for constrained submodular maximization.

It is also worth mentioning that while strong tools and techniques are developed for proving communication complexity lower bounds in this model (see, e.g. [268, 302, 67] and references therein), not much progress is made in the literature toward proving round complexity lower bounds in this model. On this front, Alon *et al.* [19] proved lower bounds for the bipartite matching problem in the setting where each machine receives the edges incident on a single vertex in the left side of the graph (this is related to the welfare maximization problem in unit-demand auctions which we elaborate more on in Section 1.3 and subsequently in Chapter 9). This lower bound was improved by Braverman and Oshman

in [73] to obtain tight bounds. However, these results focus on the regime of communication complexity $\ll n$ per machine (where n is the number of vertices in the graph) and are not applicable to the setting of parameters we consider for optimization on massive graphs in which we allow at least $\Omega(n)$ communication per machine and are hence interested in super-linear in n communication lower bounds. As was nicely pointed out by Guruswami and Onak [170], proving super-linear lower bounds requires embedding a difficult problem into the "space of edges" as opposed to the "space of vertices", which turns out to be a much more difficult task in many cases.

1.2.3. Massively Parallel Algorithms

As stated in Section 1.1.3, classical parallel algorithms for the PRAM model typically give rise to MPC algorithms without incurring any asymptotic blow up in the number of rounds. As such, one can readily translate the large body of work on PRAM algorithms to the MPC model. The interesting question here is then to exploit the additional power of the MPC model (more local storage and computational power) over PRAM to achieve algorithms with much smaller number of rounds than the ones "inherited" from the PRAM model.

The first such improvement over PRAM algorithms in context of graph optimization was achieved by Karloff *et al.* [213] who developed algorithms for graph connectivity and minimum spanning tree in $O(1)$ MPC rounds on machines with local memory $n^{1+\Omega(1)}$ (n is the number of vertices in the graph). This improves upon the $\Omega(\log n)$ rounds needed in PRAM model (and their direct MPC simulation) for these problems. Since then, numerous algorithms have been designed for various graph optimization problems including matching and vertex cover, minimum cut, densest subgraph, etc., that achieve $O(1)$ round complexity with $n^{1+\Omega(1)}$ per machine memory (see, e.g., [225, 223, 11, 9, 42] and references therein).

The next set of improvements reduced the memory per machine to $O(n)$ (possibly at the cost of a slight increase in the number of rounds). For example, an $O(1)$ round algorithm for connectivity and minimum spanning tree using only $O(n)$ memory per machine has been proposed in [197] building on previous work in [156, 178, 235] (see also [11, 55, 228] for further related results). A very recent result of [111], have also achieved an $O((\log\log n)^2)$-round algorithm for the maximum matching problem when the memory per machine is $O(n)$ or even $n/(\log\log n)^{O(\log\log(n)}$.

Alas, this progress has came to a halt at the *truly sublinear* in n regime, i.e., $n^{1-\Omega(1)}$ space per-machine. This setting of parameter is particularly relevant to *sparse* graphs with $O(n)$ edges, as in this scenario, $\Omega(n)$ memory per-machine allows to fit the entire input on a single machine, thereby trivializing the problem.

Submodular optimization and in particular constrained submodular maximization have also been extensively studied in the MPC model [99, 63, 223, 250, 186, 249, 112, 113]. For

instance, Kumar *et al.* [223] developed a method to implement the greedy algorithms for constrained submodular maximization in $O(\log m)$ MPC rounds on machines of memory $m^{\Omega(1)}$ where m is the number of items in the input. This was subsequently improved dramatically by [113] to $O(1)$ MPC rounds and asymptotically the same memory per machine. Another line of research also considered achieving algorithms with small approximation ratio for constrained submodular maximization and its special cases such as maximum coverage, in a very small number of rounds, i.e., in only one or two MPC rounds [186, 249, 112].

Finally, as pointed out already in Section 1.1.3, proving round complexity lower bounds in the MPC model turns out to be a challenging task (see, e.g., [277] for implication of such lower bounds to long standing open problems in complexity theory). As a result, most previous work on lower bounds concerns either communication complexity in a fixed number of rounds or specific classes of algorithms (for round lower bounds); see, e.g., [3, 53, 269, 189] (see also [277] for more details).

1.3. Our Main Contributions

We now presents our main contributions in this thesis and the background for each one. We start by presenting our results for **graph optimization**. Tables 1 and 2 contain, respectively, a summary of our algorithms and impossibility results in this part.

Chapter 3 – Maximum Matching in the Streaming Model. We saw in Section 1.2.1 that for nearly all graph problems studied in insertion-only streams, researchers were able to subsequently obtain algorithms with similar guarantees in the dynamic streaming model as well. A curious omission from the list of successes was the prominent problem of *maximum matching* in single-pass dynamic streams. This was quite surprising given that matching is one of the most studied problems in the graph streaming literature. Achieving algorithms for the matching problem in dynamic streams was a main open problem in this area (for example, it featured prominently in "List of Open Problems in Sublinear Algorithms" [59]).

In this thesis, we fully settle the space complexity of the maximum matching problem in single-pass dynamic streams. As a corollary of this result, we obtain that any single-pass dynamic streaming algorithm that achieves even a weak approximation ratio of $n^{o(1)}$ to the maximum matching problem requires $n^{2-o(1)}$ space (here n is the number of vertices in the graph). This is the *first* lower bound on the space complexity of any natural graph optimization problem in dynamic stream which is "easy" in insertion-only streams (maximum matching admits a simple 2-approximation in $O(n)$ space in insertion-only streams).

We further consider the (algorithmically easier) problem of estimating the size of a maximum matching (as opposed to finding the actual edges) in both insertion-only and

dynamic streams. We show that while this problem provably requires less space than finding approximate matchings in both models, achieving a near optimal solution, i.e., a $(1+\varepsilon)$-approximation, still requires (almost) quadratic in n space. As a corollary of this result, we also obtain that estimating the rank of a given matrix in data streams requires (almost) quadratic space. Our results constitute the first super-linear lower bound (in number of vertices of the graph or rows of the matrix) for both problems, addressing open questions posed by Kapralov *et al.* [205] and Li and Woodruff [232].

The materials in this chapter are based on two papers, one with Sanjeev Khanna, Yang Li, and Grigory Yaroslavtsev [35] in SODA'16, and another one with Sanjeev Khanna and Yang Li [33] in SODA'17 (also invited to "Highlights of Algorithms" conference, HALG'17).

Chapter 4 – A Framework for Graph Optimization on Massive Graphs. Incidentally, our results in [35, 33] can also rule out applicability of existing general-purpose algorithmic approaches for designing one round/pass streaming, distributed, and MPC algorithms such as linear sketching and composable coresets (discussed in Section 1.1.4) for solving matching and the closely related vertex cover problems. This suggests that to address these problems in a unified way in these models, new techniques are needed.

To address this issue, we develop a new framework for designing algorithms for graph optimization problems, in particular matching and vertex cover, in the three models above. Our main insight is that the intractability of matching and vertex cover is inherently connected to the adversarial ordering/partitioning of the underlying graph in these models. Building on this, we propose a general approach that can achieve significantly better algorithms for these problems under the assumption that the input is randomly ordered/partitioned (this assumption is *not* even needed for MPC algorithms).

We further use this approach to design a single unified algorithm that improves the state-of-the-art in all the three models studied in this thesis simultaneously. For example, in random arrival streams, our algorithm computes an (almost) 3/2-approximate matching in a single pass with $O(n^{1.5})$ space (here n is the number of vertices); this significantly improves upon previous single-pass algorithms using subquadratic space, and is the first result to present strong evidence of a separation between random and adversarial order for matching. Another example is in the MPC model: Given $O(n^{1.5})$ space per machine, our algorithm computes an (almost) 3/2-approximate matching in only two MPC rounds; this significantly improves upon all previous results with a small constant number of rounds.

This chapter is based on a paper with Sanjeev Khanna [30] in SPAA'17 (recipient of the "best paper award"; also invited to "Highlights of Algorithms", HALG'18) and a paper with Hossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein [29] in SODA'19.

Chapter 5 – Massively Parallel Algorithms for Matching and Vertex Cover with Linear Memory Per Machine. One of the most natural and practical choice of memory per machine in the MPC model when processing massive graphs is $O(n \cdot \text{polylog}(n))$, i.e., near-linear in the number of vertices (similar-in-spirit to restrictions in other models such as semi-streaming). Unfortunately however, previous *round-efficient* algorithms in the MPC model for most graph optimization problems, including matching and vertex cover, all required $n^{1+\Omega(1)}$ memory per machine. This situation was somewhat remedied very recently by a result of Czumaj *et al.* [111] in STOC'18 that achieved an $O((\log \log n)^2)$-round algorithm for the maximum matching problem on machines with memory $O(n)$, even in fact slightly sublinear in n, i.e., $O(n/\text{polylog}(n))$. The authors of [111] conjectured that the number of rounds in this result can be reduced to $O(\log \log n)$ MPC rounds and furthermore posed the question of obtaining similar algorithms for the minimum vertex cover problem as an open problem.

We build on our techniques in the previous part to design an algorithm that achieves an $O(1)$-approximation to both matching and vertex cover in only $O(\log \log n)$ MPC rounds and $O(n/\text{polylog}(n))$ memory per machine, hence settling the conjecture of [111] in affirmative, and resolving their open question regarding vertex cover. Perhaps more importantly, we achieve these improvements using a fairly simple algorithm and analysis (both the algorithms and analysis in [111] were quite complicated).

The materials in this part are based on a (solo-author) manuscript [26], and a paper with Hossein Bateni, Aaron Bernstein, Vahab Mirrokni, and Cliff Stein [29] in SODA'19.

Chapter 6 – Massively Parallel Algorithms for Connectivity on Sparse Graphs. Massive graphs that appear in practice are believed to be typically sparse, i.e., have only $O(n)$ edges, where n is the number of vertices in the graph. One natural example is social networks, in which most participants are likely to have a bounded number of friends. Despite this, most computational models for studying massive graphs including the streaming and distributed communication models are inherently not suited to process sparse graphs. The reason is that for most problems of interest, it is easy (often even trivial) to prove an $\Omega(n)$ lower bound on the space or communication complexity in these models (even for sparse graphs), and on the other hand allowing $O(n)$ space or communication "trivialize" the problem in case of sparse graphs as the whole input can be stored or communicated.

One important exception is the MPC model which *in principle* allows for a non-trivial treatment of sparse graphs as well. In particular, having an algorithm with $n^{1-\Omega(1)}$ memory per machine would allow processing massive sparse graphs in a memory efficient manner. However, to date, all existing algorithms for graph problems in the MPC model that use

$n^{1-\Omega(1)}$ memory per machine, require $\Omega(\log n)$ rounds which is prohibitively large in practice. This is also quite unfortunate from the theoretical point of view as algorithms with such performance can be achieved by simple MPC simulation of classical $O(\log n)$-round PRAM algorithms that have been known in the literature for more than three decades. In other words, *current algorithms cannot benefit from the additional power of the MPC model over PRAM algorithms (more local storage and computational power) on* ***sparse graphs***.

The research in the MPC model has identified a captivating algorithmic challenge for breaking the linear-memory barrier in the MPC model: connectivity on sparse undirected graphs. However, despite the great attention this problem has received in the last few years (see, e.g. [213, 271, 218, 23, 277] and references therein), no better than $o(\log n)$-round algorithm for this problem has yet been achieved. This led some researchers to conjecture that such algorithms may not even exist [53, 271, 23, 277, 307] and in fact use sparse graph connectivity as a hardness assumption for proving *conditional* lower bounds in the MPC model for other problems; see [23, 307] and references therein for further details (recall that by the remark at the end of Section 1.1.3, achieving even "weak" *unconditional* lower bounds in the MPC model seems beyond the reach of current techniques).

In this thesis, we take an *opportunistic* approach to the sparse connectivity problem, which exploits the *connectivity structure* of the underlying graph. In particular, we use *spectral gap* as a quantitative measure of "connectedness" of a graph and design an algorithm for connectivity with improved performance guarantee depending on the spectral gap of the connected components of the underlying graph. For example, when connected components of the graph have large spectral gap, say $\Omega(1)$ or even $\Omega(1/\text{polylog}(n))$, our algorithm only requires $O(\log\log n)$ MPC rounds while using $n^{\Omega(1)}$ memory per machine and $O(n \cdot \text{polylog}(n))$ total memory. Examples of such graphs include random graphs and expanders (see also [239, 159] for real-life examples in social networks). This constitutes the *first* non-trivial improvement on the standard $O(\log n)$-round algorithms for connectivity in the MPC model when the memory per machine is $n^{\Omega(1)}$ for a general family of input graphs.

This chapter is based on a joint work with Xiaorui Sun and Omri Weinstein [36].

We now present our results for **submodular optimization** over massive datasets. Tables 3 and 4 contain, a summary of our algorithms and impossibility results in this part.

Chapter 7 – Coverage Problems in the Streaming Model. Minimum set cover and maximum coverage, henceforth collectively referred to as coverage problems, are among the canonical examples of submodular optimization. Introduced originally by Saha and Getoor [281], streaming coverage problems have been studied extensively in the streaming literature [281, 108, 40, 128, 116, 187, 174, 90, 50, 129, 247], resulting in numerous efficient

Problem	Model	Mem./Comm.	Apx	Rounds/Passes
Maximum Matching	random streams	$O(n^{1.5})$	1.5	one pass
	random partition distributed model	$O(n)$	1.5	one round
	MPC model	$O(n^{1.5})$	1.5	two rounds
		$O(n)$	$1+\varepsilon$	$O(\log\log n)$ rounds
Minimum Vertex Cover	random partition distributed model	$O(n)$	3	one round
	MPC model	$O(n)$	$O(1)$	$O(\log\log n)$ rounds
Sparse Connectivity	MPC model	$n^{\Omega(1)}$	-	$O(\log\log n + \log\frac{1}{\lambda})$

Table 1: A sample of our algorithms for graph optimization on massive graphs. The third column measures the space in case of streaming algorithms, communication per machine in case of distributed algorithms, and memory per machine in case of MPC algorithms. Here n is the number of vertices and λ (in the last row) is a lower bound on the spectral gap of every connected component of the input graph. For simplicity of exposition, logarithmic factors are omitted in the third column (similarly $(1+\varepsilon)$ factors in the approximation ratio column when appropriate).

algorithms for these problems. However, a shortcoming of all these algorithms for the streaming set cover problem was that they either required strictly more than one pass over the stream, or achieved very large approximation ratios of $O(\sqrt{n})$ (where n is the number of elements in the universe). As such, obtaining *single-pass* streaming algorithms for set cover with sublinear space over the stream was a main open problem in this area (cf. [187] and its follow-up version in [174]).

In this thesis, we establish *tight* bounds on the space-approximation tradeoff for single-pass streaming algorithms for the set cover problem. Our results rule out the existence of any non-trivial single-pass algorithm for the streaming set cover problem and hence provide a strong negative answer to the aforementioned open questions. This also suggests a strong separation between power of single-pass vs multi-pass (even only *two*-pass) streaming algorithms for this problem: For instance, while one can achieve a 2-approximation algorithm to set cover in only two passes over the input and $O(m\sqrt{n})$ memory (with exponential running time) [116], in one pass over the stream, the best approximation ratio achievable in $O(m\sqrt{n})$ memory is $\Omega(\sqrt{n})$ (here m is the number of sets and n is the size of universe).

Problem	Model	Mem./Comm.	Apx	Rounds/Passes
Maximum Matching & Minimum Vertex Cover	dynamic streams	$n^{2-o(1)}$	$n^{o(1)}$	one pass
	distributed model	$n^{2-o(1)}$	$n^{o(1)}$	one round
	MPC model	$n^{2-o(1)}$	$n^{o(1)}$	one round
	random partition distributed model	$\Omega(n)$	$O(1)$	one round
Estimating Maximum Matching Size & Matrix Rank	insertion-only streams	$n^{1+\Omega(\frac{1}{\log\log n})}$	$1+o(1)$	one pass
	dynamic streams	$n^{2-O(\varepsilon)}$	$1+\varepsilon$	one pass
	dynamic streams	$n^{1-o(1)}$	$n^{o(1)}$	one pass

Table 2: A sample of our impossibility results for graph optimization on massive graphs. The third column measures the space in case of streaming algorithms, communication per machine in case of distributed algorithms, and memory per machine in case of MPC algorithms. Here n is the number of vertices in the graph or rows of the matrix.

We further study the space-approximation tradeoff for multi-pass streaming algorithms for both set cover and coverage problems. We (slightly) improve the state-of-the-art algorithms for set cover that are allowed multiple passes over the stream and more importantly also prove that the space-approximation tradeoff achieved by this algorithm is information-theoretically optimal. Qualitatively similar results for maximum coverage are also presented.

This chapter is based on a joint work with Sanjeev Khanna and Yang Li [32] in STOC'16, and a (solo-author) paper [27] in PODS'17 (recipient of "best student paper award").

Chapter 8 – Submodular Maximization in the Distributed Communication Model. Submodular maximization subject to cardinality constraint and its illustrative example, maximum coverage, have been studied extensively in models of computation for massive datasets including in distributed communication model (e.g., [186, 249]), MPC model (e.g., [99, 223]), and the streaming model (e.g. [50, 247]).

Previous constant factor approximation algorithms for maximum coverage and submodular maximization with cardinality constraint in general in the distributed model can be divided into two main categories: *communication efficient* protocols that require a large

number of rounds (e.g., [40, 247]), or *round efficient* protocols that incur a large communication cost (e.g. [223]). For the maximum coverage problem, these two categories amount to $\widetilde{O}(n)$ communication and $\Omega(k)$ rounds in one case, and $O(1)$ rounds and $k \cdot m^{\Omega(1)}$ communication in the other case (here m and n are respectively the number of sets and size of the universe in the input instance and k is the number of machines). This state-of-the-affairs raised the following natural question: *Does there exist a truly efficient distributed protocol submodular maximization and in particular maximum coverage, that is, a protocol that simultaneously achieves $\widetilde{O}(n)$ communication cost, $O(1)$ round complexity, and gives a constant factor approximation?*

We refute the possibility of this optimistic scenario in this thesis. We present a *tight tradeoff* between the three main measures of efficiency in this model: the approximation ratio, the communication cost, and the number of rounds for any algorithm for the maximum coverage problem. This proves that for the maximum coverage problem in particular and constrained submodular maximization in general, one is always handicapped with either a large communication cost or a large number of rounds in the distributed communication model. To prove this result, we provide a general framework for proving communication lower bounds for *bounded-round* protocols hence also partially addressing the shortcoming of previous techniques for proving round complexity lower bounds in this model (recall the discussion at the end of Section 1.2.2).

As a corollary of our results, we also obtain lower bounds on the number of passes needed to solve these two problems in dynamic streams. This is the first multi-pass lower bound for any optimization problem that is specific to dynamic streams, i.e., does not follow from a lower bound in insertion-only streams. Interestingly, these impossibility results also guides us to develop a very simple MPC algorithm for submodular maximization subject to cardinality constraint with performance guarantee that matches the state-of-the-art algorithms in the MPC model [113].

This chapter is based on a paper with Sanjeev Khanna [31] in SODA'18.

We further consider optimization in settings where the goal is to perform computation over data which may not be particularly large but still imposes restrictions of similar nature, the setting which we refer to as **resource constrained optimization**. We use the toolkit developed in the first two parts of the thesis to obtain several algorithms and impossibility results for problems of this nature, settling multiple open questions in the literature.

Chapter 9 – Interaction in Combinatorial Auctions. We study the necessity of interaction between bidders with subadditive valuations in a combinatorial auctions for maximizing the social welfare. This problem was originally introduced by Dobzinski, Nisan,

Problem	Model	Mem./Comm.	Apx	Rounds/Passes
Set Cover	insertion-only streams	$O(m \cdot n^{1/\alpha})$	α	$O(\alpha)$ passes
Maximum Coverage	dynamic streams	$O(n)$	$\frac{e}{e-1}$	$O(\log n)$ passes
	distributed model	$O(n)$	$n^{1/(r+1)}$	r rounds
	MPC model	$m^{\Omega(1)}$	$\frac{e}{e-1}$	$O(1)$ rounds
Submodular Maximization with Cardinality Constraint	dynamic streams	$O(n)$	$\frac{e}{e-1}$	$O(\log n)$ passes
	MPC model	$m^{\Omega(1)}$	$\frac{e}{e-1}$	$O(1)$ rounds

Table 3: A sample of our algorithms for submodular optimization over massive datasets. The third column measures the space in case of streaming algorithms, communication per machine in case of distributed algorithms, and memory per machine in case of MPC algorithms. Here m and n are respectively the number of sets (items) and size of the universe. For simplicity of exposition, logarithmic factors are omitted in the third column (similarly $(1+\varepsilon)$ factors in the approximation ratio column when appropriate).

and Oren [119] as the following simple market scenario: m items are to be allocated among n bidders in a distributed setting where bidders valuations are private and hence communication is needed. The communication happens in rounds: in each round, each bidder, simultaneously with others, broadcasts a message to all parties involved. The central planner computes an allocation solely based on the communicated messages.

Dobzinski *et al.* [119] showed that (at least some) interaction is necessary for obtaining any efficient allocation: no non-interactive (1-round) protocol with polynomial communication (in the number of items and bidders) can achieve approximation ratio better than $\Omega(m^{1/4})$, while $O(\log m)$ rounds of interaction suffice to obtain an (almost) efficient allocation, i.e., a polylog(m)-approximation. Subsequently, Alon, Nisan, Raz, and Weinstein [19] studied the *qualitatively similar but technically disjoint* setting of bidders with unit-demand valuations and proved that $\Omega(\log \log m)$ rounds of interaction are necessary in this case. Dobzinski *et al.* [119] and Alon *et al.* [19] both posed the problem of proving round complexity lower bounds for the setting of combinatorial auctions with subadditive valuations as an open problem. Alon *et al.* [19] mentioned that: "from a communication complexity

Problem	Model	Mem./Comm.	Apx	Rounds/Passes
Set Cover	insertion-only streams	$\Omega(m \cdot n/\alpha)$	α	one pass
		$O(\frac{1}{p} \cdot m \cdot n^{1/\alpha})$	α	p passes
Maximum Coverage	insertion-only streams	$\Omega(\frac{1}{p} \cdot m/\varepsilon^2)$	$1+\varepsilon$	p passes
	dynamic streams	$n^{\omega(1)}$	$O(1)$	$o(\log n)$ passes
	distributed model	$n^{\omega(1)}$	$o(n^{1/2r})$	r rounds

Table 4: A sample of our impossibility results for submodular optimization over massive datasets. The third column measures the space in case of streaming algorithms, communication per machine in case of distributed algorithms, and memory per machine in case of MPC algorithms. Here m and n are respectively the number of sets (items) and size of the universe. For simplicity of exposition, logarithmic factors are omitted in the third column. As maximum coverage is a special case of submodular maximization with cardinality constraint, all lower bounds for maximum coverage clearly hold for the latter problem as well.

perspective, lower bounds in this setup [of combinatorial auctions] are more compelling, since player valuations require exponentially many bits to encode, hence interaction has the potential to reduce the overall communication from exponential to polynomial".

We resolve this fascinating question by providing an almost tight *round-approximation* tradeoff for this problem, when the players are communicating only polynomially many bits (in n and m). As a corollary, we prove that $\Omega(\frac{\log m}{\log \log m})$ rounds of interaction are *necessary* for obtaining any efficient allocation (i.e., a constant or even a polylog(m)-approximation) in these markets. Our proof builds on the similarity between this problem and the distributed communication model and the lower-bound framework we provide in Chapter 8.

The materials in this chapter are based on a (solo-author) paper [25] in EC'17.

Chapter 10 – Learning With Limited Rounds of Adaptivity. In many learning settings, active/adaptive querying is possible, but the number of rounds of adaptivity—the number of rounds of interaction with the feedback generation mechanism—is limited. For example, in crowdsourcing, one can actively request feedback by sending queries to the crowd, but there is typically a waiting time before queries are answered; if the overall task

is to be completed within a certain time frame, this effectively limits the number of rounds of interaction. Similarly, in marketing applications, one can actively request feedback by sending surveys to customers, but there is typically a waiting time before survey responses are received; again, if the marketing campaign is to be completed within a certain time frame, this effectively limits the number of rounds of interaction.

We study the relationship between query complexity and adaptivity in identifying the k most biased coins among a set of n coins with unknown biases. This problem is a common abstraction of many well-studied problems, including the problem of identifying the k best arms in a stochastic multi-armed bandit, and the problem of top-k ranking from pairwise comparisons. Our main result establish an *optimal* lower bound on the number of rounds adaptivity needed to achieve the optimal worst case query complexity for all these problems. In particular, we show that, perhaps surprisingly, no constant number of rounds suffices for this task, and the "correct" number of rounds of adaptivity is $\log^*(n)$ (an upper bound of $\log^*(n)$ rounds was also established in a joint work with Agarwal, Agarwal, and Khanna [4]).

The materials in this chapter are based on a joint paper with Arpit Agarwal, Shivani Agarwal, and Sanjeev Khanna [4] in COLT'17.

1.4. How to Read this Thesis

In Chapter 2, we introduce the basic definitions and primary tools that are used throughout this thesis. In Chapters 3, 4, 5, and 6, we present the first part of our results on graph optimization. Chapters 7 and 8 contain the second part of our results for submodular optimization. The final part of our results, i.e., the applications of main tools in this thesis to other areas, appear in Chapters 9 and 10. Each of these chapters starts with the definition and significance of the addressed problems, a review of the relevant literature, followed by a formal statement of the contributions together with an overview of the proof techniques. The formal proofs are given in the main body of the chapter. These chapters are designed to be self-contained (beside assuming the background in Chapter 2) to allow the reader to directly consider the part of most interest. The only exception is Chapters 4 and 5 which are closely tied to each other and are hence better to be read in this order.

Chapter 2
Background

In this chapter, we provide the basic definitions, notation and facts that will be employed throughout this thesis. This in particular includes simple tools from theory of submodular functions, information theory, communication complexity, and information complexity. Additional notation with a specific scope are defined locally in each respective section.

2.1. Notation and Preliminaries

General notation. For any integer $t \geq 1$, we define $[t] := \{1, \ldots, t\}$. We say that a set $S \subseteq [n]$ with $|S| = s$ is a *s-subset* of $[n]$. For a k-dimensional tuple $X = (X_1, \ldots, X_k)$ and index $i \in [k]$, we define $X^{<i} := (X_1, \ldots, X_{i-1})$ and $X^{-i} := (X_1, \ldots, X_{i-1}, X_{i+1}, \ldots, X_k)$. All logarithms are in base two unless stated explicitly otherwise. The notation "$X \in_R U$" indicates that X is chosen uniformly at random from a set U.

When it may lead to confusion, we use san-serif font, e.g. A, to denote random variables, and normal font, e.g., A, to denote the value these variables take (this notation is mostly used in our information-theoretic arguments).

We use the terms "w.p." and "w.h.p." to abbreviate "with probability" and "with high probability" respectively, where with high probability means a probability which is close to one by an additive factor which is polynomially small in the dimension of the problem (e.g., $1 - 1/\text{poly}(n)$ where n is the number of vertices in the graph).

Asymptotic notation. For simplicity of exposition, we use the "soft" O- and Ω-notation, defined as $\widetilde{O}(f) := O(f) \cdot \text{polylog}(f)$ and $\widetilde{\Omega}(g) := \Omega(g)/\text{polylog}(g)$.

Graph notation. Graphs are denoted by $G(V, E)$, where V is the set of vertices and E is the set of edges. We define $V(G) := V$ and $E(G) := E$. We use n and m to denote the number of vertices and edges in the underlying graph, i.e., $n = |V(G)|$ and $m = |E(G)|$.

2.1.1. Concentration Bounds

We use the following standard versions of Chebyshev inequality and Chernoff bound (see, e.g., [123]) throughout.

Proposition 2.1.1 (Chebyshev inequality)**.** *For any random variable X,*

$$\Pr\left(|X - \mathbb{E}[X]| \geq t\right) \leq \frac{\mathbb{V}\text{ar}[X]}{t^2}.$$

Proposition 2.1.2 (Chernoff bound)**.** *Let $X_1, \ldots, X_n$ be independent random variables*

taking values in $[0,1]$ *and let* $X := \sum_{i=1}^{n} X_i$. *Then, for any* $\varepsilon \in (0,1)$,

$$\Pr\left(|X - \mathbb{E}[X]| \geq \varepsilon \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\varepsilon^2 \cdot \mathbb{E}[X]}{2}\right).$$

In some places, we work with a generalization of Chernoff bound for variables with bounded independence.

Proposition 2.1.3 (Chernoff bound with bounded independence [285])**.** *Let* $X_1, \ldots, X_n$ *be* κ*-wise independent variables taking value in* $[0,1]$ *and* $X := \sum_{i=1}^{n} X_i$. *For any* $\delta \in (0,1)$, *if* $\kappa \leq \delta^2 \cdot \mathbb{E}[X]/2$, *then,* $\Pr\left(|X - \mathbb{E}[X]| \geq \delta \cdot \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right)$.

We also need the method of bounded differences in our proofs. A function $f(x_1, \ldots, x_n)$ is called *d-Lipschitz*, iff for all $i \in [n]$, $|f(a) - f(a')| \leq d$, whenever a and a' differ only in the i-th coordinate.

Proposition 2.1.4 (Method of bounded differences; cf [123])**.** *If* f *is* d*-Lipschitz and* $X_1, \ldots, X_n$ *are independent random variables, then,*

$$\Pr\left(|f(X_1, \ldots, X_n) - \mathbb{E}[f(X_1, \ldots, X_n)]| > t\right) \leq 2 \cdot \exp\left(-\frac{2t^2}{n \cdot d^2}\right).$$

2.1.2. Balls and Bins Experiments

We use the following standard balls and bins argument in our proofs throughout this thesis.

Proposition 2.1.5 (# of Non-Empty Bins in a Balls and Bins Experiment)**.** *Consider the process of throwing* N *balls into* B *bins where* $N \leq \varepsilon \cdot B$ *for some parameter* $\varepsilon \in (0, 1/100)$ *such that each bin is chosen uniformly at random. Let* X *denote the number of non-empty bins. Then,* $\Pr\left(|X - N| \geq 2\varepsilon \cdot N\right) \leq 2 \cdot \exp\left(-\varepsilon^2 \cdot N\right)$.

Proof. Define an indicator random variable $X_i \in \{0,1\}$ for any $i \in [B]$, where $X_i = 1$ iff the i-th bin is non-empty. Clearly $X = \sum_{i=1}^{B} X_i$ denotes the number of non-empty bins. As each bin is chosen uniformly at random by a ball, we have that,

$$\mathbb{E}[X] = \sum_{i=1}^{B} \mathbb{E}[X_i] = B \cdot \left(1 - \left(1 - \frac{1}{B}\right)^N\right) \in [(1-\varepsilon) \cdot N, (1+\varepsilon) \cdot N].$$

(using the fact that $1 - x \leq e^{-x} \leq 1 - x + x^2/2$ for $x \leq 1$ and that $N/B \leq \varepsilon$)

Random variables $X_1, \ldots, X_B$ are correlated and hence not amenable to a straightforward application of Chernoff bound. We instead use the method of bounded differences in Proposition 2.1.4 to prove the concentration of X around $\mathbb{E}[X]$.

Define N *independent* random variables $Y_1, \ldots, Y_N$, where Y_i denotes the index of the bin, the i-th ball is sent to. Define $f(Y_1, \ldots, Y_N)$ as the number non-empty bins (which is clearly only a function of $Y_1, \ldots, Y_N$). We have $f(Y_1, \ldots, Y_N) = X$ and that f is clearly 1-Lipschitz as changing any Y_i can only make one more bin empty or non-empty. As such, by Proposition 2.1.4,

$$\Pr\left(|f(Y_1, \ldots, Y_N) - \mathbb{E}\left[f(Y_1, \ldots, Y_N)\right]| > \varepsilon \cdot N\right) \leq 2 \cdot \exp\left(-2\varepsilon^2 \cdot N\right).$$

As $f(Y_1, \ldots, Y_N) = X$ and $|\mathbb{E}[X] - N| \leq \varepsilon N$, we have,

$$\Pr\left(|X - \mathbb{E}[X]| \geq 2\varepsilon N\right) \leq 2 \cdot \exp\left(-2\varepsilon^2 \cdot N\right),$$

finalizing the proof. □

2.2. Matching and Vertex Cover

We provide a brief overview of simple definitions and facts about matchings and vertex covers in this section. The interested reader is referred to excellent texts by Lovasz and Plummer [237] and Schrijver [286] for more details and missing proofs.

2.2.1. Matchings

Formally, a *matching* M of an undirected graph $G(V, E)$ is any collection of edges that do not share vertices. A vertex is *matched* by a matching M if one of its incident edges belong to M and is otherwise *free*. A matching is called *perfect* iff it matches all vertices. A matching M is called *maximum* iff its size, i.e., the number of edges it contains, is largest among all matchings in G. Throughout this thesis, we use $\mathsf{MM}(G)$ to denote the size of a maximum matching in G.

A *maximal* matching M in G is a matching which is not a proper subset of any other matching in G. Clearly, any maximum matching is also maximal but the reverse is not true. However, we have the following basic fact showing that a maximal matching cannot be much smaller than a maximum matching either.

Fact 2.2.1. *Any maximal matching M of a graph G has size at least half of the size of any maximum matching in G, i.e., $|M| \geq (1/2) \cdot \mathsf{MM}(G)$.*

Augmenting paths. An important notion in matchings is that of *augmenting paths*. A path P in G whose edges are alternately inside and outside a matching M and starts and finishes in two different free vertices is called an augmenting path for M. Formally, an augmenting path P of a matching M is an odd length path $P := v_1, v_2, \ldots, v_{2k+1}$ where $(v_{2i}, v_{2i+1}) \in M$ and $(v_{2i-1}, v_{2i}) \in G \setminus M$ for any $i \in [k]$, and v_1 and v_{2k+1} are free

vertices. An *augmentation* of matching M with respect to an augmenting path P is a new matching M' obtained from M by switching the matched edges of P (with respect to M) with unmatched edges. It is clear that an augmentation increases the size of M by one.

Fact 2.2.2. *Let M be any matching in G and M^* be a maximum matching of G. Then M admits at least $(|M^*| - |M|)$ many vertex-disjoint augmenting paths.*

An immediate corollary of Fact 2.2.2 is that a matching M is maximum in G iff it admits no augmenting paths. This suggests an iterative approach for finding a maximum matching: start from any maximal matching (which can be found easily by a simple greedy algorithm) and augment it iteratively by finding augmenting paths. This strategy is behind many of the algorithms for finding maximum matchings, including the celebrated Hopcroft-Karp algorithm [181] and Micali-Vazirani algorithm [248] that achieve (currently best) running time of $O(m\sqrt{n})$ on bipartite graphs and non-bipartite graphs, respectively.

Another simple corollary of Fact 2.2.2 is a generalization of Fact 2.2.1 that states that a matching with no "short" augmenting paths is a "good" approximation of maximum matching. Formally,

Proposition 2.2.3 ([181])**.** *For any integer $k \geq 1$, any matching M with shortest augmenting path of length $2k+1$ is a $\left(\frac{k+1}{k}\right)$-approximation to maximum matching.*

Proof. Fix any maximum matching M^*. By Fact 2.2.2, M admits at least $|M^*| - |M|$ many vertex-disjoint augmenting paths. Since the length of each such path is at least $2k+1$, and since they are vertex-disjoint (and hence edge-disjoint), the total number of such path can be at most $|M|/k$ (any such path "uses" at least k edges from M). As such, $|M^*| - |M| \leq \frac{1}{k} \cdot |M|$ and hence $|M^*| \leq \left(\frac{k+1}{k}\right) \cdot |M|$. □

In a simpler form, Proposition 2.2.3 implies that any matching with no augmenting path of length $O(1/\varepsilon)$, is a $(1+\varepsilon)$-approximate matching.

2.2.2. Vertex Cover

A vertex cover of a graph $G(V, E)$ is a collection of vertices $C \subseteq V$ such that any edge in G is incident on at least one vertex of C. We use $\mathsf{VC}(G)$ throughout this thesis to denote the size of a smallest vertex cover of G.

There are many connections between matchings and vertex covers. For example, size of any maximum matching of a graph G is always at most equal to the size of a vertex cover of G (as at least one end point of any edge from the matching need to be in the vertex cover). For *bipartite* graphs, this connection is in fact *tight*:

Fact 2.2.4. *Size of a maximum matching in a bipartite graph G is always equal to the size*

of a minimum vertex cover in G, i.e., $\mathsf{MM}(G) = \mathsf{VC}(G)$.

An easy way to prove Fact 2.2.4 is to see that the standard linear programs for matching and vertex cover are dual of each other and the matching polytope on bipartite graphs always admit an integral optimal solution.

On the other hand, one can also show that size of a minimum vertex cover cannot be much larger than a maximum matching. This is because the vertices matched by any *maximal* matching M of G form a vertex cover of G (as otherwise one can add any uncovered edge to M to get a superset matching of M, contradicting the maximality). This implies the following basic fact.

Fact 2.2.5. *For any graph G, $\mathsf{MM}(G) \leq \mathsf{VC}(G) \leq 2 \cdot \mathsf{MM}(G)$.*

We note that in Fact 2.2.5, we can bound the RHS with two times the size of any maximal matching of G not necessarily a maximum matching.

We further have the following simple proposition that suggests that a matching and vertex cover can serve as a "witness" to near-optimality of each other.

Proposition 2.2.6. *Suppose M and C are respectively, a matching and a vertex cover of a graph G such that $\alpha \cdot |M| \geq |C|$; then, both M and C are α-approximation to their respective problems.*

Proof. $\mathsf{VC}(G) \underset{\text{Fact 2.2.5}}{\geq} \mathsf{MM}(G) \geq |M| \geq \frac{1}{\alpha} \cdot |V'| \geq \frac{1}{\alpha} \cdot \mathsf{VC}(G) \underset{\text{Fact 2.2.5}}{\geq} \frac{1}{\alpha} \cdot \mathsf{MM}(G)$. □

2.3. Ruzsa-Szemerédi Graphs

Several of our results in this thesis are based on a remarkable family of extremal graphs known as Ruzsa-Szemerédi graphs, introduced by Ruzsa and Szemerédi [279]. In this section, we define these graphs and provide a brief background on them.

For any graph G, a matching M of G is an *induced matching* iff for any two vertices u and v that are matched in M, if u and v are not matched to each other, then there is no edge between u and v in G.

Definition 2.1 (Ruzsa-Szemerédi graph)**.** *A graph G is an (r,t)-*Ruzsa-Szemerédi graph *(or (r,t)-RS graph for short), iff the set of edges in G consists of t pairwise disjoint induced matchings $M_1, \ldots, M_t$, each of size r.*

Figure 1 presents an example of very simple RS graph with 4 induced matchings of size 2 each, namely a $(2,4)$-RS graph.

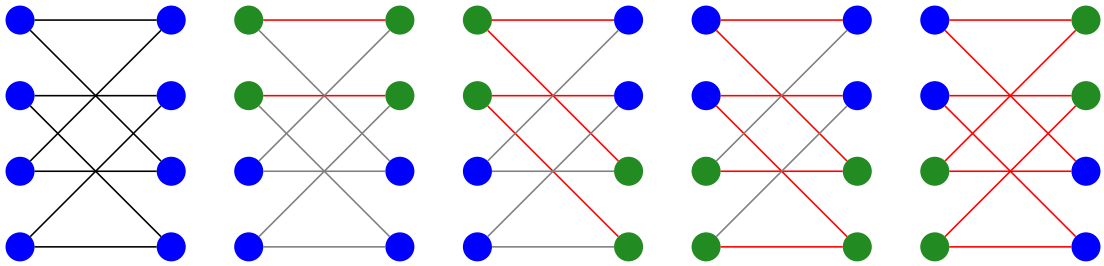


Figure 1: A $(2, 4)$-RS graph (the left most graph) along with its partitioning into induced matchings (the four right graphs).

RS graphs, first introduced by Ruzsa and Szemerédi [279], have been extensively studied as they arise naturally in property testing, PCP constructions, additive combinatorics, etc. (see, e.g., [290, 176, 143, 62, 18, 160, 16, 20, 147]). These graphs are of interest typically when r and t are large relative to the number of vertices in the graph.

One particularly interesting range of the parameters is when $r = \Theta(n)$ [143, 146, 147], i.e., when the induced matchings are of linear size. We use the notation $\mathsf{RS}(n)$ to denote the *largest* possible value for the parameter t such that an (r, t)-RS graph on n vertices with $r = \Theta(n)$ exists. It is a major open problem to determine the asymptotic of $\mathsf{RS}(n)$ [146, 163, 147], but currently there is a huge gap between existing upper and lower bounds for $\mathsf{RS}(n)$: it is known that for any constant $c < 1/4$, a $(c \cdot n, t)$-RS graph with $t = n^{\Omega(1/\log\log n)}$ exists [143] (see also [160]); however, the best known upper bound only shows that for $(c \cdot n, t)$-RS graphs, where c is any constant less than $1/4$, t is upper bounded by $\frac{n}{\log^{(x)} n}$, with $x = O(\log \frac{1}{c})$, ($\log^{(x)}(n)$ denotes the x-fold iterative logarithm of n) [146]. Slightly better upper bounds are known for large values of c; in particular, it is shown in [147] that for $1/5 < c < 1/4$, $t = O(n/\log n)$. We refer the interested reader to [147, 18] for more on the history of Ruzsa-Szemerédi graphs and to [18, 160] for their application to different areas of computer science, including proving lower bounds for streaming algorithms.

Even though obtaining (r, t)-RS graphs for $r = \Theta(n)$ and $t = n^{\Omega(1)}$ seems to be out of the scope of the current techniques, Alon *et al.* [18] provided a surprising construction of (very) dense RS graphs when we allow r to be just slightly sublinear in n:

Proposition 2.3.1 ([18])**.** *For infinitely many integers n, there are (r, t)-RS graphs on n vertices with parameters $r = n^{1-o(1)}$ and $r \cdot t = \binom{n}{2} - o(n^2)$*

2.4. Set Cover and Maximum Coverage

We define the set cover and maximum coverage problems in this section. In the set cover problem, we are given a collection of m sets $\mathcal{S} := \{S_1, \ldots, S_m\}$ from a universe $[n]$. The goal is to pick a smallest number of sets from $\mathcal{S}$ whose union is $[n]$, or in other words, *cover* the universe. The set cover problem is one of Karp's original 21 **NP**-hard problems [215]. A simple greedy algorithm that iteratively picks the set that covers the most number of uncovered elements achieves a $(\ln n)$-approximation [194, 289] and this is best possible unless $\mathbf{P} = \mathbf{NP}$ [117, 135, 238, 252].

Similarly, in the maximum coverage problem, we are given a collection of m sets $\mathcal{S} := \{S_1, \ldots, S_m\}$ from a universe $[n]$ and an integer $k \geq 1$. The goal is to pick k sets from $\mathcal{S}$ whose union has maximum size, i.e., cover the most number of elements. A simple reduction from the set cover problem establishes the **NP**-hardness of the maximum coverage problem. The greedy algorithm that that for k times, iteratively picks the set that covers the most number of uncovered elements, achieves a $\left(\frac{e}{e-1}\right)$-approximation [256] and this is also best possible unless $\mathbf{P} = \mathbf{NP}$ [135].

2.5. Submodular, XOS, and Subadditive Functions

Let $V = \{a_1, \ldots, a_m\}$ be a ground set of m items and $f : 2^V \to \mathbb{N}_+$ be a non-negative set function. Function f is called *monotone* iff $f(A) \leq f(B)$ for all $A \subseteq B \subseteq V$. When clear from the context, we abuse the notation and for $a \in V$, use $f(a)$ instead of $f(\{a\})$.

Subadditive functions. A set function $f : 2^V \to \mathbb{N}_+$ is called *subadditive* iff for all $A, B, \subseteq V$,

$$f(A \cup B) \leq f(A) + f(B). \tag{2.1}$$

XOS functions. A set function $f : 2^V \to \mathbb{N}_+$ is called *XOS* (or equivalently *fractionally subadditive*), iff there exists a family of *linear* functions $g_1, \ldots, g_t$ such that for all $A \subseteq V$:

$$f(A) = \max_{i \in [t]} g_i(A). \tag{2.2}$$

Submodular functions. A set function $f : 2^V \to \mathbb{N}_+$ is called *submodular* iff for all $A, B, \subseteq V$,

$$f(A \cup B) + f(A \cap B) \leq f(A) + f(B). \tag{2.3}$$

It is easy to see that both submodular and XOS functions are also subadditive.

In this thesis, we are mostly concerned with submodular functions. In the following, we present some of their basic properties that we use in our proofs.

2.5.1. Basic Properties of Submodular Functions

One of the main properties of submodular functions that make them attractive in many different setting is the so-called *diminishing marginal contribution* property: for any set function $f : 2^V \to \mathbb{N}_+$ and $A \subseteq V$, we define the *marginal contribution* to set A in f as a set function $f_A : 2^V \to \mathbb{N}$ such that for all $B \subseteq V$, $f_A(B) := f(A \cup B) - f(A)$. We have,

Fact 2.5.1. *A set function $f : 2^V \to \mathbb{N}_+$ is submodular iff for all sets $A \subseteq B \subseteq V$ and any item $x \in V \setminus B$: $f_A(x) \geq f_B(x)$.*

Fact 2.5.1 presents another definition of submodular functions which is equivalent to Eq (2.3).

We also use the following standard facts about monotone submodular functions.

Fact 2.5.2. *Let $f : 2^V \to \mathbb{N}_+$ be a monotone submodular function, then:*

$$\forall A \subseteq V, B \subseteq V \quad f(B) \leq f(A) + \sum_{x \in B \setminus A} f_A(x).$$

Fact 2.5.3. *Let $f : 2^V \to \mathbb{N}_+$ be a submodular function, then, for any $A \subseteq V$, $f_A(\cdot)$ is also submodular (and hence subadditive).*

2.5.2. Constrained Submodular Maximization

Let $\mathcal{I} \subseteq 2^V$ be a *hereditary* set system, that is a collection of sets such that whenever $S \in \mathcal{I}$, all subsets of S also belong to $\mathcal{I}$. An example of hereditary set systems are *matroids*. In a constrained submodular maximization problem, we are given a submodular function $f : 2^V \to \mathbb{N}_+$ and a hereditary set system $\mathcal{I}$ and the goal is to return $A^* \subseteq V$ such that: $A^* \in \arg\max_{A \in \mathcal{I}} f(A)$.

Constrained submodular maximization is studied under various constraints, i.e., hereditary set systems, including cardinality or knapsack constraints, matroids, p-systems, etc., and (near) optimal algorithms (mostly based on sequential greedy algorithms) are known for this problem under these different constraints; see, e.g. [256, 142, 85, 80, 122].

Monotone Submodular Maximization Subject to Cardinality Constraint. In this thesis, we are mostly interested in one of the most basic variants of the constraints for submodular maximization, namely the cardinality constraint: Given an integer $k \geq 1$ and a monotone submodular function $f : 2^V \to \mathbb{N}_+$, our goal is to find a set $A^* \subseteq V$ such that $|A^*| \leq k$ and $f(A^*)$ is maximized.

As coverage functions are monotone submodular, it is easy to see that submodular maximization subject to cardinality constraint generalizes the maximum coverage problem. Similar to maximum coverage problem, the greedy algorithm that that for k times, iteratively picks the item with maximum marginal contribution among the remaining items achieves a $\left(\frac{e}{e-1}\right)$-approximation and this is also best possible unless $\mathbf{P} = \mathbf{NP}$ [135].

2.6. Information Theory

We now briefly introduce some definitions and facts from information theory that are needed in this thesis. We refer the interested reader to the text by Cover and Thomas [109] for an excellent introduction to this field.

For a random variable A, we use $\textsc{supp}(\mathsf{A})$ to denote the support of A and $\textsc{dist}(\mathsf{A})$ to denote its distribution. When it is clear from the context, we may abuse the notation and use A directly instead of $\textsc{dist}(\mathsf{A})$, for example, write $A \sim \mathsf{A}$ to mean $A \sim \textsc{dist}(\mathsf{A})$, i.e., A is sampled from the distribution of random variable A.

We denote the *Shannon Entropy* of a random variable A by $\mathbb{H}(\mathsf{A})$, which is defined as:

$$\mathbb{H}(\mathsf{A}) := \sum_{A \in \textsc{supp}(\mathsf{A})} \Pr(\mathsf{A} = A) \cdot \log(1/\Pr(\mathsf{A} = A)) \tag{2.4}$$

The *conditional entropy* of A conditioned on B is denoted by $\mathbb{H}(\mathsf{A} \mid \mathsf{B})$ and defined as:

$$\mathbb{H}(\mathsf{A} \mid \mathsf{B}) := \mathop{\mathbb{E}}_{B \sim \mathsf{B}} [\mathbb{H}(\mathsf{A} \mid \mathsf{B} = B)], \tag{2.5}$$

where $\mathbb{H}(\mathsf{A} \mid \mathsf{B} = B)$ is defined in a standard way by using the distribution of A conditioned on the event $\mathsf{B} = B$ in Eq (2.4).

The *mutual information* of two random variables A and B is denoted by $\mathbb{I}(\mathsf{A}\,;\mathsf{B})$ and is defined as:

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B}) := \mathbb{H}(\mathsf{A}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}) = \mathbb{H}(\mathsf{B}) - \mathbb{H}(\mathsf{B} \mid \mathsf{A}). \tag{2.6}$$

The *conditional mutual information* $\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C})$ is $\mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C})$ and hence by linearity of expectation:

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) = \mathop{\mathbb{E}}_{C \sim \mathsf{C}} [\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C} = C)]. \tag{2.7}$$

We use H_2 to denote the binary entropy function where for any real number $0 < \delta < 1$, $H_2(\delta) = \delta \log \frac{1}{\delta} + (1-\delta) \log \frac{1}{1-\delta}$ (i.e., the entropy of a Bernoulli random variable).

2.6.1. Useful Properties of Entropy and Mutual Information

We shall use the following basic properties of entropy and mutual information throughout. Proofs of these properties mostly follow from convexity of the entropy function and Jensen's inequality and can be found in [109], Chapter 2.

Fact 2.6.1. *Let* A, B, C, *and* D *be four (possibly correlated) random variables.*

1. $0 \leq \mathbb{H}(\mathsf{A}) \leq \log|\textsc{supp}(\mathsf{A})|$. *The right equality holds iff* $\textsc{dist}(\mathsf{A})$ *is uniform.*
2. $\mathbb{I}(\mathsf{A}\,;\mathsf{B}) \geq 0$. *The equality holds iff* A *and* B *are* independent.
3. Conditioning on a random variable reduces entropy*:* $\mathbb{H}(\mathsf{A} \mid \mathsf{B},\mathsf{C}) \leq \mathbb{H}(\mathsf{A} \mid \mathsf{B})$. *The equality holds iff* $\mathsf{A} \perp \mathsf{C} \mid \mathsf{B}$.
4. Subadditivity of entropy*:* $\mathbb{H}(\mathsf{A},\mathsf{B} \mid \mathsf{C}) \leq \mathbb{H}(\mathsf{A} \mid C) + \mathbb{H}(\mathsf{B} \mid \mathsf{C})$.
5. Chain rule for entropy*:* $\mathbb{H}(\mathsf{A},\mathsf{B} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C}) + \mathbb{H}(\mathsf{B} \mid \mathsf{C},\mathsf{A})$.
6. Chain rule for mutual information*:* $\mathbb{I}(\mathsf{A},\mathsf{B}\,;\mathsf{C} \mid \mathsf{D}) = \mathbb{I}(\mathsf{A}\,;\mathsf{C} \mid \mathsf{D}) + \mathbb{I}(\mathsf{B}\,;\mathsf{C} \mid \mathsf{A},\mathsf{D})$.
7. Data processing inequality*: suppose* $f(\mathsf{A})$ *is a deterministic function of* A*, then* $\mathbb{I}(f(\mathsf{A})\,;\mathsf{B} \mid \mathsf{C}) \leq \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C})$.

The following Fano's inequality states that if a random variable A can be used to estimate the value of another random variable B, then A should "consume" most of entropy of B.

Fact 2.6.2 (Fano's inequality)**.** *Let* A,B *be random variables and* f *be a function that given* A *predicts a value for* B*. If* $\Pr(f(\mathsf{A}) \neq \mathsf{B}) \leq \delta$*, then* $\mathbb{H}(\mathsf{B} \mid \mathsf{A}) \leq H_2(\delta) + \delta \cdot (\log|\mathsf{B}| - 1)$*. If* B *is binary, then the bound improves to* $\mathbb{H}(\mathsf{B} \mid \mathsf{A}) \leq H_2(\delta)$.

We also use the following two standard propositions, regarding the effect of conditioning on mutual information.

Proposition 2.6.3. *For random variables* $\mathsf{A},\mathsf{B},\mathsf{C},\mathsf{D}$*, if* $\mathsf{A} \perp \mathsf{D} \mid \mathsf{C}$*, then,*

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \leq \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C},\mathsf{D}).$$

Proof. Since A and D are independent conditioned on C, by Fact 2.6.1-(3), $\mathbb{H}(\mathsf{A} \mid \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{D})$ and $\mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{B}) \geq \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{B},\mathsf{D})$. We have,

$$\begin{aligned}\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) &= \mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{B}) = \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{D}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{B}) \\ &\leq \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{D}) - \mathbb{H}(\mathsf{A} \mid \mathsf{C},\mathsf{B},\mathsf{D}) = \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C},\mathsf{D}).\end{aligned}$$

□

Proposition 2.6.4. *For random variables* $\mathsf{A}, \mathsf{B}, \mathsf{C}, \mathsf{D}$*, if* $\mathsf{A} \perp \mathsf{D} \mid \mathsf{B}, \mathsf{C}$*, then,*

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \geq \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}).$$

Proof. Since $\mathsf{A} \perp \mathsf{D} \mid \mathsf{B}, \mathsf{C}$, by Fact 2.6.1-(3), $\mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}) = \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}, \mathsf{D})$. Moreover, since conditioning can only reduce the entropy (again by Fact 2.6.1-(3)),

$$\begin{aligned}\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) &= \mathbb{H}(\mathsf{A} \mid \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}) \geq \mathbb{H}(\mathsf{A} \mid \mathsf{D}, \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}) \\ &= \mathbb{H}(\mathsf{A} \mid \mathsf{D}, \mathsf{C}) - \mathbb{H}(\mathsf{A} \mid \mathsf{B}, \mathsf{C}, \mathsf{D}) = \mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}, \mathsf{D}).\end{aligned}$$

□

Finally, we also use the following simple inequality that states that conditioning on a random variable can only increase the mutual information by the entropy of the conditioned variable.

Proposition 2.6.5. *For random variables* A, B *and* C*,* $\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) \leq \mathbb{I}(\mathsf{A}\,;\mathsf{B}) + \mathbb{H}(\mathsf{C})$.

Proof. By chain rule for mutual information (Fact 2.6.1-(6)), we can write:

$$\begin{aligned}\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) &= \mathbb{I}(\mathsf{A}\,;\mathsf{B}, \mathsf{C}) - \mathbb{I}(\mathsf{A}\,;\mathsf{C}) = \mathbb{I}(\mathsf{A}\,;\mathsf{B}) + \mathbb{I}(\mathsf{A}\,;\mathsf{C} \mid \mathsf{B}) - \mathbb{I}(\mathsf{A}\,;\mathsf{C}) \\ &\leq \mathbb{I}(\mathsf{A}\,;\mathsf{B}) + \mathbb{H}(\mathsf{C} \mid \mathsf{B}) \leq \mathbb{I}(\mathsf{A}\,;\mathsf{B}) + \mathbb{H}(\mathsf{C}),\end{aligned}$$

where the first two equalities are by chain rule (Fact 2.6.1-(6)), the second inequality is by definition of mutual information and its positivity (Fact 2.6.1-(2)), and the last one is because conditioning can only reduce the entropy (Fact 2.6.1-(3)). □

2.6.2. Measures of Distance Between Distributions

We use two main measures of distance (or divergence) between distributions, namely the *Kullback-Leibler divergence* (KL-divergence) and the *total variation distance*.

KL-divergence. For two distributions μ and ν over the same probability space, the *Kullback-Leibler divergence* between μ and ν is denoted by $\mathbb{D}(\mu \mid\mid \nu)$ and defined as:

$$\mathbb{D}(\mu \mid\mid \nu) := \mathop{\mathbb{E}}_{a \sim \mu}\left[\log \frac{\Pr_\mu(a)}{\Pr_\nu(a)}\right]. \tag{2.8}$$

We also have the following relation between mutual information and KL-divergence.

Fact 2.6.6. *For random variables* $\mathsf{A}, \mathsf{B}, \mathsf{C}$*,*

$$\mathbb{I}(\mathsf{A}\,;\mathsf{B} \mid \mathsf{C}) = \mathop{\mathbb{E}}_{(b,c) \sim (\mathsf{B},\mathsf{C})}\Big[\mathbb{D}(\textsc{dist}(\mathsf{A} \mid \mathsf{C} = c) \mid\mid \textsc{dist}(\mathsf{A} \mid \mathsf{B} = b, \mathsf{C} = c))\Big].$$

Total variation distance. We denote the total variation distance between two distributions μ and ν on the same support Ω by $|\mu - \nu|_{tvd}$, defined as:

$$|\mu - \nu|_{tvd} := \max_{\Omega' \subseteq \Omega} \left(\mu(\Omega') - \nu(\Omega')\right) = \frac{1}{2} \cdot \sum_{x \in \Omega} |\mu(x) - \nu(x)| \,. \tag{2.9}$$

We use the following basic properties of total variation distance.

Fact 2.6.7. *Suppose μ and ν are two distributions for $\mathcal{E}$, then,* $\Pr_{\mu}(\mathcal{E}) \leq \Pr_{\nu}(\mathcal{E}) + |\mu - \nu|_{tvd}$.

Finally, the following Pinskers' inequality bounds the total variation distance between two distributions based on their KL-divergence,

Fact 2.6.8 (Pinsker's inequality)**.** *For any distributions μ and ν,* $|\mu - \nu|_{tvd} \leq \sqrt{\frac{1}{2} \cdot \mathbb{D}(\mu \mid\mid \nu)}$.

2.7. Communication Complexity

Communication complexity constitutes one of the most useful methods for proving *unconditional* lower bounds. Throughout this thesis, we study various problems through the lens of communication complexity to provide further insight on (in)tractability of these problems in different computational models for processing massive datasets.

In this section, we review basic definitions and tools from communication complexity that we employ in this thesis. We closely follow the presentation in the text by Kushilevitz and Nisan [224] and refer the interested reader to this excellent text for more details. We start by defining the two-player communication model and then provide its extension to the multi-party setting we study in this thesis.

2.7.1. Two-Party Communication Complexity

We use standard definitions of the two-party communication model introduced by Yao [305]. Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a relation. We will consider two parties, Alice and Bob, who receive $x \in \mathcal{X}$ and $y \in \mathcal{Y}$ and their goal is to find some $z \in P(x, y)$. For non-trivial P, Alice and Bob will need to communicate with each other to compute any $P(x, y)$, and do this according to some fixed protocol π.

Communication happens in rounds. In each *round*, the protocol π determines which player should speak next. This information depend only on the bits communicated thus far, as this is the only information common to both parties. Whenever it is a party's turn to speak, the protocol π must specify what the party sends, and this must depend only on the communication thus far *and* the input of the party. Finally, π should also determine when the protocol terminates, in which case, it should output $\pi(x, y) \in P(x, y)$. For a protocol π, we use $\Pi_{x,y}$ to denote the transcript of the messages on input (x, y).

We are only interested in the amount of communication between Alice and Bob, measured in number of bits. We thus allow Alice and Bob to be computationally unbounded. The *cost* of a protocol π on input (x, y) is the number of bits communicated by π on (x, y), i.e., the length of the transcript $\Pi_{x,y}$, denoted by $|\Pi_{x,y}|$. The *communication cost* of a protocol π is the maximal cost of π over all inputs (x, y), denoted by $\|\pi\| := \max_{x,y} |\Pi_{x,y}|$.

We can now use this to define the communication complexity of a problem, or equivalently a relation P.

Definition 2.2 (Deterministic Communication Complexity)**.** *For a relation $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, the deterministic communication complexity of P, denoted by $\mathsf{D}(P)$, is the minimum communication cost of any protocol π that compute P.*

Notice that clearly, $\mathsf{D}(P) \leq \min(\log |\mathcal{X}|, \log |\mathcal{Y}|)$ as the RHS is the communication cost of a trivial protocol that simply communicates the whole input of one player to another.

We are most interested in two-party communication setting when Alice and Bob additionally have access to random coins. In particular, in a *private-coin* protocol Alice and Bob have access to infinite long random strings r_A and r_B, respectively, that are chosen independent of each other and the input, and are only known to their corresponding parties (hence the term "private"). In this case, the messages communicated by Alice are function of both her input x and the random string r_A, and similarly the messages communicated by Bob are function of both y and r_B.

In a *public-coin* protocol, Alice and Bob in addition to private coins, also have access to a *shared* source of randomness r_{pub} (again chosen independent of all other variables) and can use this to determine the next messages as well as the order the parties should speak. It is easy to see that a public-coin protocol is simply a distribution on private coins protocols, run by first using shared randomness to sample a random string r and then running the corresponding private coin protocol π_r.

The communication cost of a randomized protocol is the worst case number of bits sent by the protocol over any inputs (x, y) and random bits (r_A, r_B, r_{pub}) (it is also possible to define the communication cost as an average number of bits, although we do not take this approach in this thesis).

When considering a randomized protocol, we allow the protocol to output a wrong answer with some small probability. In particular, we say that a protocol π computes a relation P with δ-error iff for every (x, y): $\Pr_{r_A,r_B,r_{pub}}\left(\pi(x, y) \in P(x, y)\right) \geq 1 - \delta$.

We can now define the randomized communication complexity of a problem.

Definition 2.3 (Randomized Communication Complexity)**.** *For a relation $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$,*

and any $\delta \in (0,1)$, the randomized communication complexity of P with error probability δ, denoted by $\mathsf{R}_\delta(P)$, is the minimum communication cost of any δ-error protocol π for P that has access to public-coins.

Throughout this thesis, by communication complexity, we refer to the randomized communication complexity with some small error bounded away from half, say, $1/10$.

In Definition 2.3, we allowed the protocols to have access to public-coins. Clearly, any protocol that only uses private-coins can be simulated by a public-coin protocol by simply partitioning the public random bits between Alice and Bob so that each player have his or her "private" random coins. A close converse to this also holds due to Newman [257].

Proposition 2.7.1 ([257])**.** *Fix any $\delta, \delta' > 0$. Let π be a δ-error public-coin protocol for a relation $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ with $m := \max\{\log|\mathcal{X}|, \log|\mathcal{Y}|\}$. There exists a private-coin $(\delta+\delta')$-error protocol for P with communication cost $\|\pi'\| \leq \|\pi\| + O(\log m + \log(1/\delta'))$.*

Proposition 2.7.1 implies that we might as well just focus on public-coin protocols (as in our Definition 2.3) and use this proposition to extend the results to private-coin protocols.

Up until now, we have been discussing protocols which, for every input (x, y), err with probability at most δ, where the probability is only over the random strings of the protocol. In some scenarios, it is more convinent to look at protocols which err on a certain distribution on inputs.

Definition 2.4 (Distributional Communication Complexity)**.** *Let μ be a probability distribution over $\mathcal{X}\times\mathcal{Y}$. For a relation $P : \mathcal{X}\times\mathcal{Y} \to \mathcal{Z}$, and any $\delta \in (0,1)$, the (μ,δ)-distributional communication complexity of P, denoted by $\mathsf{D}_{\mu,\delta}(P)$, is the minimum cost of any deterministic protocol for P which errs with probability at most δ on inputs chosen from μ (the probability of error is with respect to distribution μ).*

The famous Yao's minimax principle [306] relates $\mathsf{D}_{\mu,\delta}(P)$ to $\mathsf{R}_\delta(P)$, that is:

$$\mathsf{R}_\delta(P) = \max_\mu \mathsf{D}_{\mu,\delta}(P),$$

where μ is taken over all distributions over $\mathcal{X} \times \mathcal{Y}$. For our purpose, we only need the easy direction of Yao's minimax principle, as we show below.

Proposition 2.7.2 (Easy Direction of Yao's Minimax Principle [306])**.** *Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be any relation and μ be a probability distribution over $\mathcal{X} \times \mathcal{Y}$. For any $\delta \in (0,1)$:*

$$\mathsf{R}_\delta(P) \geq \mathsf{D}_{\mu,\delta}(P).$$

Proof. Let π be any δ-error protocol for P with communication cost $\|\pi\| = \mathsf{R}_\delta(P)$. We use $r := (r_A, r_B, r_{pub})$ to denote all the randomness used by π. By definition, for all (x, y), $\Pr_r(\pi(x, y) \notin P(x, y)) \leq \delta$. As such,

$$\delta \geq \Pr_{r,\mu}(\pi(x, y) \notin P(x, y)) = \mathop{\mathbb{E}}_{r}\left[\Pr_{\mu}(\pi(x, y) \notin P(x, y))\right].$$

This means that there exists a fixed choice of r such that the protocol π_r (i.e., the protocol obtained by fixing the randomness of π to be r) errs with probability at most δ on inputs chosen from μ. But π_r is a deterministic protocol with the communication cost at most as large as π. Hence, $\mathsf{D}_{\mu,\delta}(P) \leq \|\pi_r\| \leq \|\pi\| \leq \mathsf{R}_\delta(P)$. □

One-Way Protocols

In many of the applications we consider, we have an even simpler model for communication, the *one-way communication model*. In this model, Alice and Bob are given inputs (x, y) as before, but the communication only happens in one round, in which Alice computes some function $M(x)$ of her input x and send it to Bob. Bob then attempts to compute $P(x, y)$ using only $M(x)$ and y. We have,

Definition 2.5 (One-way Communication Complexity)**.** *For a relation $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$, and any $\delta \in (0, 1)$, the one-way randomized communication complexity of P with error probability δ, denoted by $\mathsf{R}^{\text{1-way}}_\delta(P)$, is the minimum communication cost of any δ-error protocol π for P in which only a single message is sent from Alice to Bob.*

All previous definitions can be tailored similarly to the one-way communication model in a straightforward way. Moreover, it is easy to see that Proposition 2.7.2 continues to hold even for one-way protocols.

2.7.2. Some Standard Communication Problems

Throughout this thesis, we establish various communication complexity lower bounds for optimization problems we study, such as maximum matching or set cover. Some of these proofs involve a direct or indirect reduction from a well-known communication problem. For convenience of the reader, we gather these communication problems in this section, along with their communication complexity lower bounds that we use. We note that however, depending on the application, we may need a more specialized versions of these problems and their corresponding communication bounds which we would define locally later.

The following three problems are relevant to the one-way communication model.

Index problem. In the *index problem*, denoted by Index_n, Alice is given a string $x \in \{0, 1\}^n$ and Bob is given $i \in [n]$. The goal of the players is to determine the value of x_i, i.e.,

$\mathsf{Index}_n(x,i) = x_i$. It is clear that in the two-player communication model, only $O(\log n)$ communication is needed to solve the problem by Bob sending his index to Alice. The non-trivial (yet not hard to prove) fact is that in the one-way communication model, where only Alice can send a single message to Bob, $\Omega(n)$ communication is needed. Formally,

Proposition 2.7.3 ([1, 222]). *For any $n \geq 1$ and $\delta < 1/3$, $\mathsf{R}^{\text{1-way}}_{\delta}(\mathsf{Index}_n) = \Omega(n)$.*

It is immediate that this is also tight as Alice can send her input to Bob.

Boolean hidden matching problem. In the *boolean hidden matching problem (BHM)*, denoted by BHM_n, Alice is given a boolean vector $x \in \{0,1\}^n$ where $n = 2k$ (for some integer $k \geq 1$) and Bob gets a *perfect matching* M on n vertices, and a boolean vector $w \in \{0,1\}^{n/2}$. Let Mx denote the length $n/2$ boolean vector $(x_{u_1} \oplus x_{v_1}, \ldots, x_{u_{n/2}} \oplus x_{v_{n/2}})$ where $(u_1, v_1), \ldots, (u_{n/2}, v_{n/2})$ are the edges of M ($\oplus$ stands for addition modulo 2, i.e., the 2-bit XOR function). It is promised that either $Mx = w$ or $Mx = \overline{w}$. The goal of the problem is for Bob to output Yes when $Mx = w$ and No when $Mx = \overline{w}$.

The BHM problem was originally introduced by Gavinsky *et al.* [153] to prove certain separation between quantum and classical communication complexity. For our purpose, we need the following one-way communication complexity lower bound for this problem.

Proposition 2.7.4 ([153]). *For any even $n \geq 2$ and $\delta < 1/3$, $\mathsf{R}^{\text{1-way}}_{\delta}(\mathsf{BHM}_n) = \Omega(\sqrt{n})$.*

It is easy to prove a matching $O(\sqrt{n})$ upper bound using Birthday paradox.

Boolean hidden hypermatching problem. In the *boolean hidden hypermatching problem (BHH)*, denoted by $\mathsf{BHH}_{n,t}$, Alice is given a boolean vector $x \in \{0,1\}^n$ where $n = 2kt$ (for some integer $k \geq 1$) and Bob gets a *perfect t-hypermatching* M on n vertices, and a vector $w \in \{0,1\}^{n/t}$. Let Mx be the boolean vector $(\bigoplus_{1 \leq i \leq t} x_{M_{1,i}}, \ldots, \bigoplus_{1 \leq i \leq t} x_{M_{n/t,i}})$ of length n/t where $\{M_{1,1,}, \ldots, M_{1,t}\}, \ldots, \{M_{n/t,1}, \ldots, M_{n/t,t}\}$ are the edges of M (here $\oplus$ stands for addition modulo 2, i.e., the t-bit XOR function). It is promised that either $Mx = w$ or $Mx = \overline{w}$. The goal of the problem is for Bob to output Yes when $Mx = w$ and No when $Mx = \overline{w}$

It is easy to see that the boolean hidden matching problem is a special case of this problem when $t = 2$, i.e., $\mathsf{BHM}_n = \mathsf{BHH}_{n,t}$. The boolean hidden hypermatching problem was introduced by Verbin and Yu [295] for proving streaming lower bounds for various problems in single-pass streams.

Proposition 2.7.5 ([295]). *For any $t \geq 2$, integer $n = 2kt$ for some $k > 0$, and $\delta < 1/3$, $\mathsf{R}^{\text{1-way}}_{\delta}(\mathsf{BHH}_{n,t}) = \Omega(n^{1-1/t})$.*

One can prove tightness of this bound similar to Proposition 2.7.4.

All communication problems introduced above are "easy" when we consider protocols that communicate more than one round. The following two problems are "hard" in the most general two-party communication model where there is no limit on the number of rounds of communication.

Set disjointness problem. In the *Set Disjointness problem*, denoted by Disj_n, Alice and Bob are given $A, B \subseteq [n]$ and the goal is to output Yes if $A \cap B = \emptyset$ and No otherwise.

A first lower bound of $\Omega(\sqrt{n})$ for disjointness was proved by Babai *et al.* [39]. This bound was strengthened to $\Omega(n)$ by Kalyanasundaram and Schnitger [201], simplified by the Razbarov [272], and further simplified (through the notion of information complexity) by Bar-Yossef *et al.* [44], leading to the following result.

Proposition 2.7.6 ([201, 272, 44])**.** *For any $n \geq 1$ and $\delta < 1/3$, $\mathsf{R}_\delta(\mathsf{Disj}_n) = \Omega(n)$.*

We remark that further advances allowed the calculation of the communication complexity of disjointness precisely, up to additive $o(n)$ terms [68], or the correct dependence on the advantage in probability of success over random guessing [72].

Gap hamming distance problem. In the *gap hamming distance* problem, denoted by GHD_n, Alice and Bob are given two sets $A, B \subseteq [n]$ and the goal is to output:

$$\mathsf{GHD}_n(A,B) := \begin{cases} \mathsf{Yes} & \Delta(A,B) \geq n/2 + \sqrt{n} \\ \mathsf{No} & \Delta(A,B) \leq n/2 + \sqrt{n}\,, \\ \star & \text{otherwise} \end{cases} \tag{2.10}$$

where $\star$ means that the answer can be arbitrary; here, $\Delta(A,B)$ denotes the hamming distance between A and B, i.e., the size of the symmetric difference of A and B.

The gap hamming distance was originally introduced by Indyk and Woodruff [188] for proving streaming lower bounds and has since been studied extensively in the literature (see [88] and references therein), leading to the following result.

Proposition 2.7.7 ([88])**.** *For any $n \geq 1$ and sufficiently small $\delta > 0$, $\mathsf{R}_\delta(\mathsf{GHD}_n) = \Omega(n)$.*

2.7.3. Multi-Party Communication Complexity

We now extend the previous definitions for two-party communication model to the multi-party setting. The model we consider here is referred in the literature to as the multi-party number-in-hand communication model with shared blackboard.

Let $P : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathcal{Z}$ be a k-ary relation. In this model, k parties $P^{(1)}, \ldots, P^{(k)}$ receive inputs $x_1, \ldots, x_k$ and their goal is to compute $z \in P(x_1, \ldots, x_k)$. In order to do so, the parties need to communicate with each other. The communication proceeds in rounds. In each round r, the players simultaneously write a message on a shared blackboard visible to all parties. In a deterministic protocol, the message sent by any player $P^{(i)}$ in each round can only depend on the private input of the player, i.e., x_i, plus the messages of all players in previous rounds, i.e., the content of the blackboard. In a randomized protocol, we further allow the players to have access to both public and private randomness and the message of players can depend on them as well.

For a protocol π, we use $\Pi := (\Pi_1, \ldots, \Pi_k)$ to denote the transcript of the messages communicated by the parties, i.e., the content of the blackboard. In addition to the k parties, there exists an additional party called the referee or the coordinator which does not have any input and is responsible for outputting the answer in the last round, solely based on the content of the blackboard Π (and the public randomness).

The communication cost of a protocol π, denoted by $\|\pi\|$, is the sum of worst-case length of the messages communicated by players, i.e., $\|\pi\| = \sum_{i=1}^{k} |\Pi_i|$, where $|\Pi_i|$ is worst-case length of the message communicated by player $P^{(i)}$ over all choice of input and randomness. Using this, we can define the communication complexity of a problem P as follows.

Definition 2.6 (Multi-Party Randomized Communication Complexity)**.** *For a relation $P : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathcal{Z}$, integer $r \geq 1$, and any $\delta \in (0, 1)$, the r-round randomized communication complexity of P with error probability δ, denoted by $\mathsf{R}_{\delta}^{\text{r-round}}(P)$, is the minimum communication cost of any δ-error r-round protocol π for P.*

We further consider *per-player communication complexity* which informally speaking measures the maximum amount of communication done by any one player as opposed to collective communication of all players. It is clear that per-player communication complexity of a k-party problem is at least $1/k$ fraction of its communication complexity.

One can define distributional communication complexity for r-round protocols in this model the same way as before. Moreover, it is straightforward to verify that Proposition 2.7.2 continues to hold for any number of rounds and parties.

Simultaneous Protocols

A special case of the multi-party communication model that we frequently revisit in this thesis, is when only one round of communication happens, i.e., the *simultaneous communication model*. In this case, the parties receive $x_1, \ldots, x_k$ as before, but each party $P^{(i)}$ now simply computes some function $M_i(x_i)$ of the input and send it directly to the referee or the coordinator. The referee then outputs the answer using only $M_1(x_1), \ldots, M_k(x_k)$ (and

the public randomness). We refer to 1-round protocols as *simultaneous* protocols.

Definition 2.7 (Multi-Party Simultaneous Communication Complexity)**.** *For a relation $P : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathcal{Z}$, and any $\delta \in (0,1)$, the simultaneous randomized communication complexity of P with error probability δ, denoted by $\mathsf{R}^{||}_{\delta}(P)$, is the minimum communication cost of any δ-error simultaneous protocol π for P.*

One can see that for any P and δ, $\mathsf{R}^{||}_{\delta}(P) = \mathsf{R}^{\text{1-round}}_{\delta}(P)$, as the role of blackboard in 1-round protocols is unnecessary and can be ignored. As before, we can tailor all previous definitions for the simultanous communication model in a straightforward way.

Remark 2.7.8. *To facilitate our proofs throughout this thesis, when studying simultaneous protocols, we sometimes need to give the referee an auxiliary input as well, which is jointly distributed with the input of the k players. The referee's answer then would be a function of the k messages he receives as well as his input.*

2.7.4. Applications to Streaming Lower Bounds

The similarity between the setting of (multi-party) communication complexity and the distributed communication model we introduced in Section 1.1.2 is obvious and need no further explanation. This allows us to prove lower bounds on the communication cost of distributed protocols by studying their communication complexity. However, both models of two-party and multi-party communication models introduced in this section are also extremely useful for studying space complexity of streaming algorithms as we elaborate more in this section.

Two-Party Communication Complexity and Insertion-Only Streams

Many connections between two-party communication complexity and space complexity of streaming algorithms in insertion-only streams are by now standard facts and have been studied since one of the earliest papers that introduced the streaming model [17]. For completeness, we present two of such results that we use in this thesis.

Proposition 2.7.9 (Folklore; e.g. [17, 167])**.** *Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a relation. For any integer $p \geq 1$ and $\delta > 0$, any p-pass streaming algorithm that computes P on streams $A \in \mathcal{X} \circ \mathcal{Y}$ (where $\circ$ denotes the concatenation operator on two consecutive parts of the stream) with probability at least $1 - \delta$ requires $\Omega(\frac{1}{p}) \cdot \mathsf{R}_{\delta}(P)$ space.*

Proof. Let ALG be any p-pass streaming algorithm for P on streams in $\mathcal{X} \circ \mathcal{Y}$ with space complexity s. We create the following two-party protocol π for $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ from ALG:

Alice runs ALG on her input x and send the content of the memory of the streaming

algorithm to Bob as the message. As ALG is a streaming algorithm, Bob can continue running ALG on his input y using only the memory content, and again send the memory content back to Alice. This way, the players can simulate ALG on the stream $x \circ y$ using $O(p \cdot s)$ bits of communication as the memory content of ALG never has size more than s and they only need $O(p)$ back and forth communication.

This results in a $O(p \cdot s)$ communication protocol for P with the same error probability as ALG, hence $s \geq \Omega(\frac{1}{p}) \cdot \mathsf{R}_{\delta}(P)$ by definition. □

Proposition 2.7.10 (Folklore; e.g. [17])**.** *Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a relation. For any integer $p \geq 1$ and $\delta > 0$, any single-pass streaming algorithm that computes P on streams $A \in \mathcal{X} \circ \mathcal{Y}$ (where $\circ$ denotes the concatenation operator on two consecutive parts of the stream) with probability at least $1 - \delta$ requires $\mathsf{R}_{\delta}^{\text{1-way}}(P)$ space.*

Proof. The proof is identical to that of Proposition 2.7.9 using the fact that a single-pass streaming algorithm would result in a one-way protocol for P. □

Multi-Party Communication Complexity and Dynamic Streams

A by far more non-trivial connection between communication complexity and streaming model was established in two beautiful results by Li *et al.* [230] and Ai *et al.* [14] by providing an interesting characterization of dynamic streaming algorithms. We elaborate more on this connection below.

All known algorithms for problems in dynamic streams are linear sketching algorithms (recall the definition of linear sketching algorithms from Section 1.1.4). It was shown in [230] that this is not just a coincidence: any single-pass streaming algorithm for computing (approximately) any arbitrary function on multiplicity vector f_A of a dynamic stream A can be reduced to an algorithm which, before the stream begins, samples a matrix M uniformly at random from a set of hardwired integer matrices, and then maintains the linear sketch $M \cdot f_A \pmod{q}$, where $q = (q_1, \ldots, q_r)$ is a vector of positive integers and r is the number of rows of A. The space complexity of this linear sketching algorithm is only larger from the original algorithm by an additive factor of the space required to sample M and q (which is shown to be logarithmic in the dimension of the vector f in [230]).

It is a well-known fact that any linear sketching algorithm that requires at most p passes of adaptive sketching can be implemented in the multi-party communication model with p rounds of communication: each player simply computes the linear sketches on its input and writes that on the shared blackboard; by linearity of the sketches, the players can then combine these sketches and obtain a linear sketch of the whole input. This allows the players to implement each round of adaptive sketching in one round of communication and compute

the final answer. It is also easy to see that the *per player* communication cost of this new algorithm is at most the size of the linear sketch. Combining this with the reduction of [14] implies that if one can prove a lower bound on the per-player communication complexity of a problem in the multi-party communication model, one also obtains a lower bound on the space complexity of dynamic streaming algorithms; notice that since in the communication model we can perform the sampling of sketching matrix M and q via public-coins, hence free of communication charge, we do not even need to pay for the extra additive factor in space in the reduction; we refer the interested reader to [14] for more details.

Proposition 2.7.11 ([230, 14])**.** *Suppose that a p-pass randomized streaming algorithm* ALG *solves a problem* P *on any dynamic stream* A *with probability at least* $1-\delta$*, and that the space complexity of* ALG *depends only on the dimension of the frequency vector of the stream* A *(and in particular not the length of the stream). Then, for any integer* $k \geq 1$*, assuming* $P_k : \mathcal{X}_1 \times \ldots \times \mathcal{X}_k \to \mathcal{Z}$ *is a relation such that* $P_k(x_1, \ldots, x_k) = P(f_{x_1 \circ \ldots \circ x_k})$ *(where* $\circ$ *denotes the concatenation operator on two consecutive parts of the stream), the space complexity of* ALG*, denoted by* $s(\mathsf{ALG})$*, is at least as large as the per-player p-round communication complexity of* P_k *with error* 2δ*. In particular,*

$$s(\mathsf{ALG}) = \Omega\left(\frac{1}{p \cdot k}\right) \cdot \mathsf{R}_{2\delta}^{\text{p-round}}(P_k).$$

A few remarks are in order. Firstly, we assume that the problem P in Proposition 2.7.11 is only a function of the frequency vector of the stream A and not the order in which the items are being deleted or inserted. This is consistent with our definition of dynamic streams and all problems we consider in this thesis satisfy this requirement, i.e., they are only a function of the final state of the stream. Second, we assume that the space complexity of streaming algorithm for P does not have a dependence on the length of the stream. Again, this is a common assumption satisfied by all algorithms in the dynamic streaming model for optimization problems; for instance, for graph problems, the space complexity of algorithms is only a function of number of vertices, n, independent of the length of the stream. Finally, we note that while originally [14] states this result only for constant p, it continues to hold even when p is super-constant depending on the dimension of the frequency vector [300] (regardless, the main application of this result for us is the case $p = 1$, i.e., between single-pass algorithms and simultaneous communication complexity).

2.8. Information Complexity

Information complexity has emerged as an amazingly strong tool for studying various fundamental questions in communication complexity. This section introduces some of the central concepts from information complexity used in this thesis. For a more detailed overview of

information complexity, we refer the reader to [66, 297].

2.8.1. Two-Party Information Complexity

Consider the same setting of two-party communication model introduced in Section 2.7.1. Rather than measuring the communication cost of a protocol π, we are now interested in measuring the "information" revealed by this protocol, which informally speaking captures the (average) amount of additional information that Alice and Bob learn about each others inputs from the protocol π. We formalize this in the following definition.

Definition 2.8 (Internal Information Cost [89, 44, 46])**.** *The internal information cost of a protocol π on inputs chosen according to some distribution μ on $\mathcal{X} \times \mathcal{Y}$ is given by:*

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) := \mathbb{I}(\mathsf{X}\ ; \Pi, \mathsf{R}_{\mathrm{pub}} \mid \mathsf{Y}) + \mathbb{I}(\mathsf{Y}\ ; \Pi, \mathsf{R}_{\mathrm{pub}} \mid \mathsf{X}).$$

Here $(\mathsf{X}, \mathsf{Y}) \sim \mu$ denotes the joint input of Alice and Bob, Π is the random variable for the transcript of the protocol, and $\mathsf{R}_{\mathrm{pub}}$ denotes the public *coins.*

Two remarks are in order. Firstly, the information cost of a protocol π depends on the prior distribution μ, as the mutual information between the transcript Π of the protocol and the inputs depends on the prior distribution on the inputs. Secondly, we only consider public coins in the cost of the protocol *not* the private coins (informally speaking, only public coins can reveal information to the other player after conditioning on the messages).

The above notion of information cost is defined as "internal" information cost as it measures the amount of information revealed to players (who already know some information about the other player's input as the input maybe correlated in general). Another useful notion of information cost is "external" information cost which informally speaking measures the amount of information revealed to an external observer who can only sees the transcript of the protocol and not the input to players. Formally,

Definition 2.9 (External Information Cost [89, 44, 46])**.** *The external information cost of a protocol π on inputs chosen according to some distribution μ on $\mathcal{X} \times \mathcal{Y}$ is given by:*

$$\mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) := \mathbb{I}(\mathsf{X}, \mathsf{Y}\ ; \Pi, \mathsf{R}_{\mathrm{pub}}).$$

Here $(\mathsf{X}, \mathsf{Y}) \sim \mu$ denotes the joint input of Alice and Bob, Π is the random variable for the transcript of the protocol, and $\mathsf{R}_{\mathrm{pub}}$ denotes the public *coins.*

This latter definition is more useful to us when we are working with *one-way* protocols.

Remark 2.8.1. *In the context of information complexity, we always consider protocols that*

have access to ***both private- and public-coins even against a prior distribution on inputs****. This may seem counterintuitive at first glance: in randomized communication complexity, we could ignore private-coins and simply use public-coins to simulate them; similarly in distributional setting, by the easy direction of Yao's minimax principle (Proposition 2.7.2) we could ignore all randomness by fixing the randomness of the protocol to one particular choice of random bits. These arguments fail however when one considers information complexity. The reason is that it is plausible that private-coins are useful in "hiding" information about the input and hence switching them with public-coins or fixing them in the distributional setting would in principle dramatically increase the information cost of the protocol (see [297] for a simple example when this actually happens).*

Before we move on, we present a simple proposition that allows us to simplify the role of public-coins in measuring the information costs of the protocols.

Proposition 2.8.2. *For any protocol π and distribution μ on $\mathcal{X} \times \mathcal{Y}$:*

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{Y}, \mathsf{R}_{\mathrm{pub}}) + \mathbb{I}(\mathsf{Y}\,;\Pi \mid \mathsf{X}, \mathsf{R}_{\mathrm{pub}}), \qquad \mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}).$$

Proof. We prove this for external information cost; the same exact argument applied to each term in internal information cost proves the first part as well. By definition,

$$\begin{aligned}\mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi,\mathsf{R}_{\mathrm{pub}}) &\underset{\text{Fact 2.6.1-(6)}}{=} \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\mathsf{R}_{\mathrm{pub}}) + \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}) \\ &= 0 + \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}),\end{aligned}$$

where the last equality is because $\mathsf{R}_{\mathrm{pub}} \perp \mathsf{X},\mathsf{Y}$ and by Fact 2.6.1-(2). □

As any bit communicated by players can never reveal more than one bit of information, the communication cost of a protocol always upper bounds it information cost.

Proposition 2.8.3 (cf. [74])**.** *For any protocol π and any distribution μ on $\mathcal{X} \times \mathcal{Y}$:*

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) \leq \mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) \leq \|\pi\|.$$

Proof Sketch. For intuition, we prove the right inequality in above proposition:

$$\begin{aligned}\mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) &\underset{\text{Proposition 2.8.2}}{=} \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}) \leq \mathbb{H}(\Pi \mid \mathsf{R}_{\mathrm{pub}}) \\ &\underset{\text{Fact 2.6.1-(3)}}{\leq} \mathbb{H}(\Pi) \underset{\text{Fact 2.6.1-(1)}}{\leq} \log|\textsc{supp}(\Pi)| \leq \|\pi\|.\end{aligned}$$

□

One can now use the information cost of protocols to define information complexity of a problem (with respect to a distribution μ) by taking the minimum information cost of a δ-error protocol for this problem on μ. We avoid presenting this definition formally due to some subtle technical issues that need to be addressed which lead to multiple different but similar-in-spirit definitions[1]. As such, we only use the term information complexity in informal discussions and in formal statements always use information cost directly.

Proposition 2.8.3 provides us with a strong tool for proving communication complexity lower bounds by lower bounding the information cost of the protocols instead of their communication cost. This approach turns out to be quite useful as information cost exhibits "nicer" properties than communication cost in general (we give an example in next section).

We finish this section by remarking that in the one-way model, the information cost of protocols can be simplified further as the messages do not depend on Bob's input.

Proposition 2.8.4. *For any* one-way *protocol* π *on distribution* μ *over* $\mathcal{X} \times \mathcal{Y}$*:*

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{Y}, \mathsf{R}_{\mathrm{pub}}), \qquad \mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}).$$

Proof. By Proposition 2.8.2, we have,

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{Y}, \mathsf{R}_{\mathrm{pub}}) + \mathbb{I}(\mathsf{Y}\,;\Pi \mid \mathsf{X}, \mathsf{R}_{\mathrm{pub}}) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{Y}, \mathsf{R}_{\mathrm{pub}}),$$
$$\mathsf{IC}^{\mathrm{ext}}_{\mu}(\pi) = \mathbb{I}(\mathsf{X},\mathsf{Y}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}) \underset{\text{Fact 2.6.1-(6)}}{=} \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}) + \mathbb{I}(\mathsf{Y}\,;\Pi \mid \mathsf{X}, \mathsf{R}_{\mathrm{pub}}) = \mathbb{I}(\mathsf{X}\,;\Pi \mid \mathsf{R}_{\mathrm{pub}}),$$

where in both equations we used the fact that $\mathbb{I}(\mathsf{Y}\,;\Pi \mid \mathsf{X}, \mathsf{R}_{\mathrm{pub}}) = 0$ because π is a one-way protocol and hence Π is only a function of X and $\mathsf{R}_{\mathrm{pub}}$ independent of Y (and hence by Fact 2.6.1-(2) its conditional mutual information with Y is zero). □

2.8.2. Direct Sum and Additivity of Information Cost

An important notion in complexity theory is that of *direct sum*. Informally speaking, direct sum results assert a lower bound on the complexity of solving n copies of a problem P in parallel, in terms of the cost of a single copy. Let us define this more formally in the context of communication complexity.

Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be any relation and $P^n : \mathcal{X}^n \times \mathcal{Y}^n \to \mathcal{Z}^n$ denote the relation which maps the tuple $((x_1, \ldots, x_n), (y_1, \ldots, y_n))$ to $(P(x_1, y_1), \ldots, P(x_n, y_n))$. How "harder" is to solve P^n compared to P? For instance, can we relate, say, the deterministic communication complexity of P^n, i.e., $\mathsf{D}(P^n)$, to that of P, i.e., $\mathsf{D}(P)$? Certainly $\mathsf{D}(P^n) \leq n \cdot \mathsf{D}(P)$ as we

[1] An example of such issue: does the protocol need to be δ-error only on the distribution μ or is it δ-error on all inputs and only the information cost is measured over μ? Both definitions make sense and depending on the application one may need to work with one or another as we shall do in different places of this thesis.

can simply run the best protocol for P on every coordinates of P^n; can we do any better?

Generally speaking, direct sum results aim to show that this naive protocol is essentially the best possible, i.e., the cost of solving n copies of P grows linearly with n. To quote [297]: "The value of such a result is clear: a direct sum result, together with a lower bound on the (easier-to-reason-about) sub-problem, yields a lower bound on the composite problem in a "black-box" fashion (a method also known as hardness amplification)".

Reviewing the vast literature on direct sum results (even within the scope of communication complexity) is far out of the scope of the current thesis; for this, we refer the interested reader to [211, 134, 210, 89, 21, 224, 219, 46, 190, 75, 151, 297] and references therein. We only mention in passing that a recent breakthrough result of Ganor *et al.* [151] (see also [270] for a simplified proof) rules out the existence of a *tight* direct sum result for randomized communication complexity (see [151] for details).

Perhaps the single most remarkable property of information cost is that it is a fully additive measure over composition of tasks; in other words, information complexity, unlike communication complexity, indeed admits a tight direct sum result.

Proposition 2.8.5 (cf. [44, 46, 74, 297])**.** *Let $P : \mathcal{X} \times \mathcal{Y} \to \mathcal{Z}$ be a relation, μ be a distribution on $\mathcal{X} \times \mathcal{Y}$ and μ^n be the distribution on $\mathcal{X}^n \times \mathcal{Y}^n$ obtained by picking each coordinate* independently *from μ. Suppose π^n is a δ-error protocol for P^n on μ^n for $\delta > 0$. Then, there exists a δ-error protocol π for P on μ such that $\|\pi\| = \|\pi^n\|$ but*

$$\mathsf{IC}^{\mathrm{int}}_{\mu}(\pi) \leq \frac{1}{n} \cdot \mathsf{IC}^{\mathrm{int}}_{\mu^n}(\pi^n).$$

A word of interpretation is in order. In Proposition 2.8.5, we use the protocol π^n for P^n to achieve a protocol π for P with the same error probability and communication cost. This part is not at all interesting on its own; of course a protocol that solves n copies of P, i.e., P^n can also solve one copy of it with the same cost and error probability. The interesting part however is that this new protocol has a much smaller information cost than the original protocol, i.e., by a factor of n. Hence, we obtain a protocol for P which communicates "a lot" while revealing a "tiny" amount of information (on average) about the input. Using this we can easily prove a direct sum result for information complexity: if solving P requires protocols with internal information cost at least I, then solving P^n, i.e., n parallel copies of P, requires at least $n \cdot I$ information cost, i.e., n times larger. Even though we do not use this proposition directly in this thesis, we use many different simple or not-so-simple extensions of it tailored to the specific problem at hand to prove various information complexity and ultimately communication complexity lower bounds.

Part I

Graph Optimization on Massive Graphs

Chapter 3
Maximum Matching in the Streaming Model

Starting from this chapter, we delve into details of our own contributions in this thesis and present our results and techniques developed along the way to prove these results. We begin with our impossibility results for maximum matching problem in the streaming model in this chapter. Throughout this chapter we solely focus on single-pass streaming algorithms. The materials in this chapter are based on two papers [35, 33] with additional simplifications and details along with some previously unpublished results.

As stated earlier in Section 1.2.1, graph optimization constitutes one of the most active area of research on streaming algorithms. In particular, the maximum matching problem, the focus of this chapter, has been studied extensively in this model [241, 139, 125, 130, 160, 221, 311, 11, 8, 170, 204, 205, 110, 102, 242, 9, 131, 220, 35, 101, 83, 244, 267, 33, 29]. Despite this, several fundamental questions regarding the complexity of this problem in graph streams have remained unresolved. In particular,

(i) *How well can we approximate the maximum matching problem in a single-pass over a dynamic streams? (cf. the "List of Open Problems in Sublinear Algorithms" [59])*

A straightforward 2-approximation single-pass algorithm in $O(n)$ space over insertion-only streams has been known for over a decade (by computing a maximal matching). However, no non-trivial single-pass streaming algorithm using space $o(n^2)$ was previously known. This is in sharp contrast to most other graph problems that admit algorithms with similar guarantees in both insertion-only and dynamic streams (as discussed in Section 1.2.1).

(ii) *Can we estimate the* ***size*** *of a maximum matching in a space strictly smaller than what is needed for finding edges of an approximate matching? In general, what is the space-approximation tradeoff for estimating the maximum matching size in graph streams? (cf. [131, 244, 205])*

Previously, it was known that achieving better than $e/(e-1)$-approximation to maximum matching size requires $n^{\Omega(1/\log\log n)}$ [204, 160, 205], while getting $(1+\varepsilon)$-approximation requires up to $n^{1-O(\varepsilon)}$ space [131, 83] (these bounds hold for both insertion-only and dynamic streams). No super-linear in n lower bounds on space or super-constant lower bounds on approximation were known previously. On the other hand, the only existing non-trivial algorithm is a folklore that an $O(\sqrt{n})$-approximation can be obtained in $\text{polylog}(n)$ space even in dynamic streams. We note that other algorithms that use $o(n)$ space for this problem also exist, but they

only work under certain conditions on the input: either the edges are presented in a *random order* [205] or the input graph has *bounded arboricity* [131, 101, 83, 244].

We address these fundamental questions from both upper bound and lower bound ends in this chapter. In particular, we resolve the space-approximation tradeoff for maximum matching in single-pass over dynamic streams, hence fully settling the first open question above. Our results suggest that to even achieve a very weak approximation ratio of $n^{o(1)}$ to the maximum matching problem, an almost quadratic space of $n^{2-o(1)}$ is needed, which is in sharp contrast to complexity of this problem in insertion-only streams.

We further provide a host of new upper and lower bounds on space complexity of estimating the size of a maximum matching in both insertion-only and dynamic streams. Our results show that while the problem of matching size estimation is provably easier than the problem of finding an approximate matching, the space complexity of the two problems starts to converge together rapidly as the accuracy desired in the computation approaches near-optimality. In particular, we establish both the first super-linear space lower bound (in number of vertices) and the first super-constant approximation lower bound for the matching size estimation problem. A well-known connection between matching size and matrix rank allows us to carry our lower bound results to the problem of estimating rank of a matrix in the streaming model, and we show that essentially quadratic space is necessary to obtain a near-optimal approximation of matrix rank.

HighLights of Our Contributions

In this chapter, we will establish:

- Tight upper and lower bounds on the space needed to find an approximate matching in dynamic streams (Sections 3.4 and 3.5).
- Lower bounds on the space needed to estimate maximum matching size to within very large approximation factors in dynamic streams (Section 3.6).
- Lower bounds on the space needed to estimate maximum matching size very accurately in both insertion-only streams and dynamic streams (Section 3.7).

Along the way, we also discuss further applications of our impossibility results to other models of computation such as MPC and distributed communication model.

3.1. Background

Recall that in an insertion-only streams, the edges of an input (multi-)graph are presented one by one in a stream, while in dynamic streams, the stream contains both insertions and deletions of an edges of a (multi-)graph and the goal is to solve the problem on the set of edges present at the end of the stream.

Maximum matching is among the most well-studied problems in the graph streaming literature. For single-pass insertion-only streams, it is known that any exact algorithm for the maximum matching problem (even for matching size problem) requires $\Omega(n^2)$ space [139]. It is also easy to compute a 2-approximate matching using $\widetilde{O}(n)$ space in insertion-only streams: simply maintain a *maximal* matching during the stream; here n denotes the number of vertices in the input graph. This can be done similarly for computing an α-approximate matching in $\widetilde{O}(n/\alpha)$ space for any $\alpha \geq 2$. These are the best $o(n^2)$ space algorithms known for the maximum matching problem. On the lower bound side, it is shown in [204, 160] that computing better than a $e/(e-1)$-approximate matching requires $n^{1+\Omega(1/\log\log n)}$ space.

Despite the huge body of work on the matching problem in insertion-only streams, for dynamic graph streams, no non-trivial single-pass streaming algorithm using space $o(n^2)$ was known previously. Indeed, the only previous result concerning matchings in the single-pass dynamic graph streams is the recent paper by Chitnis *et al.* [102], which provides an algorithm for computing a maximal matching of size k using $\widetilde{O}(nk)$ space. However, for multi-pass dynamic graph streams, Ahn and Guha [9] provide a $(1+\varepsilon)$-approximation algorithm for the weighted non-bipartite matching problem using $O(p/\varepsilon)$ passes with $\widetilde{O}(n^{1+1/p})$ space (see also [242]).

For the seemingly easier problem of estimating the maximum matching size, the result of [204, 160] can be modified to show that computing better than a $e/(e-1)$-approximation for matching size requires $n^{\Omega(1/\log\log n)}$ space (see also [205]). It was shown later in [131] that computing better than a 3/2-approximation requires $\Omega(\sqrt{n})$ bits of space. More recently, this lower bound was extended by [83] to show that computing a $(1+\varepsilon)$-estimation requires $n^{1-O(\varepsilon)}$ space. On the other hand, the only existing non-trivial algorithm is a folklore that an $O(\sqrt{n})$-approximation can be obtained in polylog(n) space even in dynamic streams (see [33] Appendix A). We note that other algorithms that use $o(n)$ space for this problem also exist, but they only work under certain conditions on the input: either the edges are presented in a *random order* [205] or the graph has *bounded arboricity* [131, 101, 83, 244].

3.2. Our Results and Techniques

We say that an algorithm α-approximates the maximum matching problem iff it finds an actual matching (i.e., the set of edges) which is within an α factor of the maximum matching. An algorithm is said to α-estimates the maximum matching size, iff it outputs a number which is an α-approximation to the size of the maximum matching. In the following, we present our results for α-approximation and α-estimation separately.

3.2.1. Approximation Algorithms for the Maximum Matching Problem

We resolve the space complexity of dynamic streaming algorithms for approximating maximum matchings by proving tight upper and lower bounds on their space requirement.

Result 3.1. *There exists a single-pass streaming algorithm that outputs an α-approximate matching w.h.p. in dynamic graph streams, while using (i) $\widetilde{O}(n^2/\alpha^3)$ space for $\alpha \leq \sqrt{n}$ and (ii) $\widetilde{O}(n/\alpha)$ space for $\alpha \geq \sqrt{n}$.*

Our algorithm in Result 3.1 is a *sampling based* algorithm that takes advantage of the well-known ℓ_0-samplers in dynamic stream. The algorithm maintains a set of (edge) samplers that are coordinated in such a way that the sampled edges are "well-spread" across different parts of the graph, and hence contain a large matching. We point out that for weighted graphs with poly(n)-bounded weights, the Crouch-Stubbs technique [110] can be used to obtain a similar result for approximating weighted matchings, while increasing the space complexity by a factor of $O(\log n)$.

Note that $\Omega(n/\alpha)$ is always a lower bound on the space requirement of any α-approximation algorithm for matching simply since the output can be this large whenever the optimal matching is of size $\Omega(n)$. Therefore, our algorithm immediately achieves optimal space requirement (up to log-factors) when $\alpha \geq \sqrt{n}$. More interestingly, we also show a space lower bound of $\Omega(n^2/\alpha^3)$ for $\alpha \leq \sqrt{n}$. Since $n^2/\alpha^3 \geq n/\alpha$ for any $\alpha \leq \sqrt{n}$, our Result 3.1 achieves the optimal space bound (up to log-factors) for any approximation ratio.

Result 3.2. *For any $\alpha \leq \sqrt{n}$, any randomized single-pass streaming algorithm that outputs an α-approximate matching in dynamic graph streams with a constant probability requires $\Omega(n^2/\alpha^3)$ space in the worst case.*

As a corollary of Result 3.2 (by setting $\alpha = n^{o(1)}$), we obtain that even a very weak approximation ratio of $n^{o(1)}$ for maximum matching problem requires almost-quadratic space of $n^{2-o(1)}$ (recall that $O(n^2)$ is a trivial upper bound on the space complexity of any graph problem in graph streams). This resolves an open problem posed at the Bertinoro workshop on sub-linear and streaming algorithms in 2014 (see the "List of Open Problems in Sublinear Algorithms" [59]), regarding the possibility of having a constant factor approximation algorithm for the maximum matching in sub-quadratic space in dynamic streams.

As was shown previously in Proposition 2.7.11, the results in [230, 14] imply that to establish Result 3.2, it suffices to prove the same lower bound on the per-player communication complexity of simultaneous protocols for the maximum matching problem. We establish such a lower bound for simultaneous protocols following the line of work by [160, 204] on using constructions based on Ruzsa-Szemerédi graphs (RS graphs) [279], which are graphs that can be decomposed into large-size induced matchings (recall their exact definition from

Section 2.3). However, our focus is on simultaneous protocols (instead of one-way protocols studied previously) and polynomial approximation regime (instead of constant). We provide a new approach for this setting that benefits from a very dense construction of RS graphs [18] and hence bypass the $n^{1+\Omega(1/\log\log n)}$ barrier in the aforementioned work on the value of the space lower bound.

Remark 3.2.1. *Our Result 3.2 improves upon our previous published result in [35] (see also [34]) by a factor of $n^{o(1)}$ in the space complexity, at a cost of increasing multiplicity of edges in the underlying graph by a factor of $n^{o(1)}$.*

We formalize Results 3.1 and 3.2 in Sections 3.4 and 3.5, respectively.

3.2.2. Estimation Algorithms for the Maximum Matching Size Problem

One can prove that α-estimating maximum matching size is strictly easier than finding an α-approximate matching: There exist single-pass streaming algorithms that for any $2 \leq \alpha \leq \sqrt{n}$, w.h.p., output an α-estimation of the maximum matching size in insertion-only streams using $\widetilde{O}(n/\alpha^2)$ space and in dynamic streams using $\widetilde{O}(n^2/\alpha^4)$ space, respectively [33]. This provides a non-trivial separation between approximate estimation and approximate computation of matchings in both dynamic and insertion-only streams, which in turn suggests that to prove lower bounds for the estimation problem, new ideas are needed as we do in this part.

Our first lower bound result concerns computing an α-approximation of the maximum matching size in dynamic streams for any $\alpha \geq 1$, not necessarily a constant. Recall that a graph G has arboricity ν if the set of edges in G can be partitioned into at most ν forests.

Result 3.3. *Any randomized single-pass streaming algorithm that computes an α-estimation of maximum matching size with a constant probability in dynamic streams requires $\Omega(\sqrt{n}/\alpha^{2.5})$ bits of space. This bound holds even if the input graph is both* sparse *and has* arboricity *$O(\alpha)$. Moreover, if the input graph is allowed to be* dense*, then $\Omega(n/\alpha^2)$ bits of space is necessary.*

The lower bounds in Result 3.3 are the first non-trivial space lower bounds for *super-constant* approximation algorithms for matching size estimation. Obtaining space lower bounds for polylog(n)-approximation of matching size has been posed as an open problem by Kapralov *et al.* [205], who also mentioned that "existing techniques do not seem to lend easily to answer this question and it will be very useful (quite possibly for other related problems) to develop tools needed to make progress on this front". Our Result 3.3 makes progress on this question in dynamic streams.

An interesting aspect of our lower bound in Result 3.3 is that it holds even for bounded

arboricity graphs. There is an active line of research on estimating matching size of bounded arboricity graphs in graph streams [101, 83, 131, 244], initiated by Esfandiari *et al.* [131]. The state-of-the-art (at the time our results were established) is an $O(1)$-approximation in $\widetilde{O}(n^{4/5})$ space for dynamic streams in bounded-arboricity graphs [101, 83, 244] (see Section 3.2.3 for subsequent results).

Our second lower bound result concerns computing a $(1+\varepsilon)$-approximation of the maximum matching size in both insertion-only streams and in dynamic streams. In the following, let $\mathsf{RS}(n)$ denote the maximum number of edge-disjoint *induced matchings* of size $\Theta(n)$ in any n-vertex graph (recall the definition of $\mathsf{RS}(n)$ from Section 2.3).

Result 3.4. *Any (randomized) single-pass streaming algorithm that with a constant probability outputs a $(1+\varepsilon)$-estimation of the maximum matching size in insertion-only streams requires $\mathsf{RS}(n) \cdot n^{1-O(\varepsilon)}$ space. The lower bound improves to $n^{2-O(\varepsilon)}$ for dynamic streams.*

Since $\mathsf{RS}(n)$ is known to be at least $n^{\Omega(1/\log\log n)}$ [143], Result 3.4 immediately implies that no $\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$-space algorithm can output a $(1+\varepsilon)$-approximation of matching size in insertion-only streams. Interestingly, it is known that by allowing multiple passes over the stream, a $(1+\varepsilon)$-approximate matching (as opposed to only its size) can be found in $\widetilde{O}(n \cdot \mathrm{poly}(1/\varepsilon))$ space, even in dynamic streams and even for the weighted version of the problem [9, 8] (see also [242]).

Our lower bounds in Result 3.4 are the first super linear (in n) space lower bounds for estimating matching size in graph streams. An interesting implication of these lower bounds is that while the problem of matching size estimation is provably easier than the problem of finding an approximate matching (by our results in [33]), the space complexity of the two problems starts to converge together rapidly as the accuracy desired in the computation approaches near-optimality.

Schatten p-norms. The *Schatten p-norm* of a matrix A is defined as the ℓ_p-norm of the vector of the singular values of A (see [232] for more detail); in particular, the case of $p=0$ corresponds to the *rank* of the matrix A. Schatten norms and rank computation have been previously studied in the streaming and sketching models [104, 83, 232, 229, 233, 231]. It is shown that exact computation of matrix rank in data streams requires $\Omega(n^2)$ space [104, 231] (even allowing multiple passes), and $(1+\varepsilon)$-approximation requires $n^{1-O(\varepsilon)}$ space [83]; the latter result was recently extended to all Schatten p-norms for *odd* values of p [232].

It is well-known that computing the maximum matching size is equivalent to computing the rank of the Tutte matrix [292, 237]. Consequently, all our lower bounds stated for matching size estimation also hold for matrix rank computation. This in particular implies

an $\Omega(\sqrt{n})$ space lower bound for *any constant* approximation of rank in *sparse* matrices and a near-optimal $n^{2-O(\varepsilon)}$ space lower bound for $(1+\varepsilon)$-approximation in *dense* matrices, answering an open question of Li and Woodruff [232].

3.2.3. Subsequent Work

Approximation Algorithms. Independently and concurrently to our work on approximating matchings in dynamic graph streams published in [35] (see [34] for an earlier version of our work), Chitnis *et al.* [101] and Konrad [220] also obtained new results on this problem. Chitnis *et al.* [101] also developed an α-approximation algorithm in dynamic graph streams using $\widetilde{O}(n^2/\alpha^3)$ space for $\alpha \leq \sqrt{n}$ (similar to first part of our Result 3.1). Although both the algorithm of [101] and our algorithm are based on sampling, our result is somewhat stronger in that (i) we also obtain an optimal space bound for the regime $\alpha > \sqrt{n}$, and (ii) we achieve an update time of $\text{polylog}(n)$ in contrast to an update time of $O(n)$ achieved by [101]. Konrad [220] gives an upper bound of $\widetilde{O}(n^2/\alpha^2)$ on space for α-approximation algorithm and a lower bound of $\Omega(n^{3/2}/\alpha^4)$. Both our upper and lower bound results (Results 3.1 and 3.2) are stronger than the results established in [220]. We point out that while at a high level, the lower bound approach used in our work and the one used in [220] are similar, the constructions and proof techniques are quite different.

Estimation Algorithms. After our work on estimating maximum matching size was published in [33], new algorithms for estimating the maximum matching size in bounded arboricity graphs were developed in [107, 245], culminating in an $O(\nu)$-approximation to the size of maximum matching in graphs with arboricity ν in only $O(\log n)$ space in insertion-only streams, and $\widetilde{O}(n^{2/3})$ space in dynamic streams. Note that our Result 3.3 (first part) rules out $o(\nu)$-approximation in $o(\sqrt{n})$ space in graph with arboricity ν in dynamic streams.

3.3. Preliminaries

ℓ_0-Samplers. We use the following powerful tool developed in the streaming literature for performing sampling in a dynamic stream.

Definition 3.1 (ℓ_0-sampler [148])**.** *An ℓ_0-sampler is an algorithm which given access to a dynamic stream A with a d-dimensional frequency vector f_A, either outputs FAIL or outputs an index $i \in [d]$, where i is chosen uniformly at random from the **support** of f_A.*

We use ℓ_0-samplers to recover an edge between a pre-defined set of vertices, if one exists.

Lemma 3.3.1 ([195])**.** *For any $0 < \delta < 1$, there is an implementation of ℓ_0-sampler for vectors in $\mathbb{R}^n$, which fails with probability at most δ, using $O(\log^2 n \cdot \log(\delta^{-1}))$ bits of space.*

3.3.1. Boolean Hidden Hypermatching Problem

Recall the boolean hidden hypermatching problem (BHH) from Section 2.7.2. In this problem, Alice is given a string $x \in \{0,1\}^n$, Bob is given a t-hypermatching M on vertices $[n]$ and a vector $w \in \{0,1\}^{n/t}$, and needs to determine whether $M \cdot x = w \pmod 2$ or $M \cdot x = \overline{w} \pmod 2$. The following is a hard distribution for $\mathsf{BHH}_{n,t}$ used in [295]:

The distribution $\mathcal{D}$ for $\mathsf{BHH}_{n,t}$.

- **Alice**: The input to Alice is a vector $x \in \{0,1\}^n$ chosen *uniformly at random*.
- **Bob**: The input to Bob is a perfect t-hypermatching M chosen *uniformly at random* and a boolean vector w such that, w.p. $1/2$, $w = Mx$ and w.p. $1/2$, $w = \overline{Mx}$.

Proposition 3.3.2 (Distributional Communication Complexity of $\mathsf{BHH}_{n,t}$ [295])**.** *For any $t \geq 2$, suppose $n = 2kt$ for some integer $k \geq 1$, and $\delta \in (0, 1/2)$. Let $\gamma := \frac{1}{2} - \delta$; then,*

$$\mathsf{D}^{\text{1-way}}_{\mathcal{D},\delta}(\mathsf{BHH}_{n,t}) = \Omega\left(\gamma \cdot n^{1-1/t}\right).$$

We point out that the communication lower bound for $\mathsf{BHH}_{n,t}$ stated in [295] (and similarly for BHM_n stated in [153]), has a dependence of γ^2 instead of γ; however, obtaining the linear dependence on γ is straightforward and we omit the details (see our paper [33]).

For our application, we need a (stronger) lower bound on the *information complexity* of $\mathsf{BHH}_{n,t}$ rather than its communication complexity. This result follows from those of [295, 153] with proper modifications. One way to prove this result is to use the message compression technique of [191] for bounded-round communication protocols (as generally speaking, while there is an exponential separation between information complexity and communication complexity [151, 152, 270], for bounded-round communication protocols it can be shown that these two measures are asymptotically the same (ignoring the dependence on error probability)); see also our paper [33] for a self-contained proof.

Proposition 3.3.3 (Information Complexity of $\mathsf{BHH}_{n,t}$)**.** *For any $t \geq 2$, any $n = 2kt$ for some integer $k \geq 1$, and any constant $\delta < 1/2$, the information cost of any δ-error* one-way *protocol π for $\mathsf{BHH}_{n,t}$ on distribution $\mathcal{D}$ is $\mathsf{IC}^{\text{ext}}_{\mathcal{D}}(\pi) = \Omega\left(n^{1-1/t}\right)$.*

BHH and matching size estimation. For our purpose, it is more convenient to work with a special case of the $\mathsf{BHH}_{n,t}$ problem, namely $\mathsf{BHH}^0_{n,t}$ where the vector $w = 0^{n/t}$ and hence the goal of Bob is simply to decide whether $Mx = 0^{n/t}$ (Yes case) or $Mx = 1^{n/t}$ (No case). We define $\mathsf{BHM}^0_n := \mathsf{BHH}^0_{n,2}$ (similar to BHM_n; see Section 2.7.2). It is known that (see, e.g. [295, 83, 232]) any instance of the original $\mathsf{BHH}_{n,t}$ problem can be reduced to an

instance of $\mathsf{BHH}^0_{2n,t}$ *deterministically* and with *no communication* between the players.

Corollary 3.5. *For any $n = 2kt$ (for some integer $k \geq 1$), there exists a distribution $\mathcal{D}_{\mathsf{BHH}}$ for $\mathsf{BHH}^0_{n,t}$ such that:*

- *For any $\delta \in (0,1)$ and $\gamma := \frac{1}{2} - \delta$, $\mathsf{D}^{\text{1-way}}_{\mathcal{D}_{\mathsf{BHH}},\delta}(\mathsf{BHH}^0_{n,t}) = \Omega(\gamma \cdot n^{1-1/t})$.*
- *For any* constant *$\delta < 1/2$, and any δ-error one-way protocol π for $\mathsf{BHH}^0_{n,t}$ on distribution $\mathcal{D}_{\mathsf{BHH}}$, $\mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{BHH}}}(\pi) = \Omega(n^{1-1/t})$.*
- *Alice's input $\mathsf{X} \sim \mathcal{D}_{\mathsf{BHH}}$ is supported on boolean vectors $x \in \{0,1\}^n$ with $\|x\|_0 = \frac{n}{2}$.*

The $\mathsf{BHH}^0_{n,t}$ problem has been used previously in [131, 83] to prove lower bounds for estimating maximum matching size in data streams. We briefly describe this connection.

The following reduction was first proposed by [83]. Given an instance $(x, \mathcal{M})$[1] of $\mathsf{BHH}^0_{n,t}$, we create a graph $G(V \cup W, E)$ with $|V| = |W| = n$ as follows:

- For any $x_i = 1$, Alice adds an edge between v_i and w_i to E.
- For any hyperedge $\boldsymbol{e}$ in the t-hypermatching $\mathcal{M}$, Bob adds to E a clique between the vertices w_i where i is incident on $\boldsymbol{e}$.

The following claim, proven originally by [83], establishes the correctness of this reduction. For the sake of completeness, we provide a simple proof this claim here.

Claim 3.3.4 ([83])**.** *Suppose $G(V \cup W, E)$ is the graph obtained from an instance $(x, \mathcal{M})$ of $\mathsf{BHH}^0_{n,t}$ (for an* even integer *t) with the property that $\|x\|_0 = n/2$;*

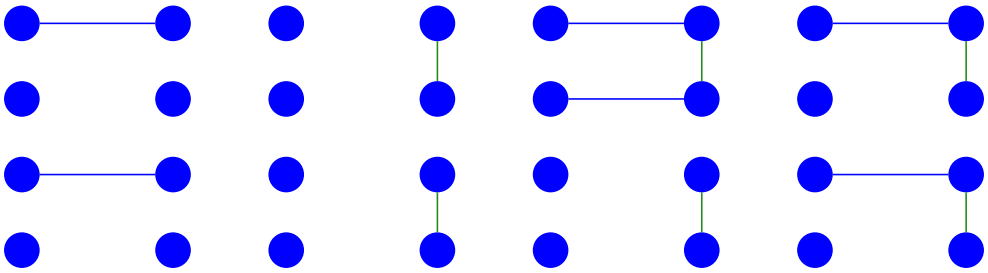
- *if $Mx = 0^{n/t}$ (i.e., Yes case), then $\mathsf{MM}(G) = \frac{3n}{4}$,*
- *if $Mx = 1^{n/t}$ (i.e., No case), then $\mathsf{MM}(G) = \frac{3n}{4} - \frac{n}{2t}$.*

Recall that $\mathsf{MM}(G)$ denotes the maximum matching size in G.

Proof. Denote by M^* a maximum matching in G. Since the vertices in V all have degree one, without loss of generality, we can assume all edges in $V \times W$ belong to M^*, and we only need to consider the maximum matching size between the remaining vertices. Since the remaining vertices in V all have degree 0, we only need to consider the remaining vertices in W (and $n/2$ vertices in W remains since $\|x\|_0 = \frac{n}{2}$).

In the Yes case, for each hyperedge $\boldsymbol{e}$, the clique created by $\boldsymbol{e}$ has t vertices, and even number of these vertices will be matched by edges in $V \times W$. Since t is even, *even* number of the vertices of the clique remain. Since there is still a clique between these remaining

[1] In order to distinguish between matchings and hypermatchings, when not clear from the context, we use $\mathcal{M}$ instead of M to denote a hypermatching.



(a) Alice's input. (b) Bob's input. (c) Yes case. (d) No case.

Figure 2: Illustration of the reduction in Claim 3.3.4 when $t = 2$.

vertices, there is a matching that matches all of them, and hence has size $\frac{n}{2} + \frac{1}{2} \cdot \frac{n}{2} = \frac{3n}{4}$.

In the No case, for each hyeredge $\boldsymbol{e}$, the clique created by $\boldsymbol{e}$ has *odd* number of vertices remained. Therefore, for every hyperedge, one vertex will be left unmatched. Since there are $\frac{n}{t}$ hyperedges, $\mathsf{MM}(G) \leq \frac{n}{2} + \frac{1}{2}\left(\frac{n}{2} - \frac{n}{t}\right) = \frac{3n}{4} - \frac{n}{2t}$. See Figure 2 for an illustration. □

3.4. An α-Approximation Algorithm for Matching

In this section, we establish the following theorem, formalizing Result 3.1.

Theorem 3.6. *There exists a single-pass streaming algorithm that outputs an α-approximate matching w.h.p. in dynamic graph streams, while using (i) $\widetilde{O}(n^2/\alpha^3)$ space for $\alpha \leq \sqrt{n}$ and (ii) $\widetilde{O}(n/\alpha)$ space for $\alpha \geq \sqrt{n}$. Moreover, the algorithm has* polylog(n) *update time for each edge insertion/deletion.*

Without loss of generality, we make the following assumptions. First, we assume that the input graph is bipartite; otherwise by applying the standard technique of choosing a random bipartition of the vertices upfront (using a pairwise independent hash function) and only considering edges that cross the bipartition, we can make the graph bipartite, while increasing the approximation ratio by a factor of 2. Moreover, we assume that the algorithm is provided with a value $\widetilde{\text{opt}}$ that is a 2-approximation of opt := $\mathsf{MM}(G)$, i.e., the size of a maximum matching in G. This is without loss of generality, since we can run our algorithm for $O(\log n)$ different estimates of opt in parallel and output the largest matching among the matchings found for all estimates. We further assume our goal is to output an $O(\alpha)$-approximation not exactly an α-approximation; a simple rescaling of parameters extend the result to latter case as well. Finally, to simplify the analysis, we can assume opt $\geq 10^3 \cdot \alpha$, since otherwise a single edge is an $O(\alpha)$-approximation of the maximum matching, which can be obtained by maintaining an ℓ_0-sampler over all edges in the graph.

At a high level, our algorithm randomly (using pairwise independent hash functions)

partitions the vertices on each side into $\Theta(\text{opt}/\alpha)$ groups. Then, for each group on the left, it chooses a subset of $\widetilde{O}(\text{opt}/\alpha^2)$ groups on the right uniformly at random and maintain one ℓ_0-sampler between the left group and each chosen group on the right. At the end of the stream the algorithm samples one edge from each ℓ_0-sampler and computes a maximum matching of these edges. Formally,

DynamicStreamMatching$(G, \widetilde{\text{opt}})$**.** A single-pass dynamic stream algorithm for computing an α-approximate matching.

Input: A dynamic graph stream defining a bipartite graph $G(L, R, E)$ with n vertices on each side, and a 2-approximation $\widetilde{\text{opt}}$ of the maximum matching size $\text{opt} := \mathsf{MM}(G)$.
Output: A matching M with size $\Omega(\text{opt}/\alpha)$.

- **Pre-processing:**
  1. Let $\gamma := \left\lceil \frac{\widetilde{\text{opt}}}{\alpha} \right\rceil$ and $\beta := 100 \left\lceil \frac{\widetilde{\text{opt}}}{\alpha^2} \right\rceil \cdot \log n$.
  2. Create two collections $\mathcal{L}$ and $\mathcal{R}$, each containing γ sets (called *groups*). Create two *pairwise independent* hash functions $h_L : L \mapsto \mathcal{L}$ and $h_R : R \mapsto \mathcal{R}$. Each vertex $u \in L$ (resp. $v \in R$) is assigned to the group $h_L(u) \in \mathcal{L}$ (resp. $h_R(v) \in \mathcal{R}$).
  3. For each $L_i \in \mathcal{L}$, assign β groups in $\mathcal{R}$ to L_i chosen *independently and uniformly at random* with replacement. For each R_j assigned to L_i, we say R_j is an *active partner* of L_i and (L_i, R_j) form an *active pair*.
- **Streaming updates:**
  - For each active pair (L_i, R_j), maintain an ℓ_0-sampler over the edges between the vertices assigned to L_i and R_j.
- **Post-processing:**
  - Compute a maximum matching over the edges sampled from the ℓ_0-samplers.

We first note that in the following, whenever we use ℓ_0-samplers, we always apply Lemma 3.3.1 with parameter $\delta = n^{-3}$. Since the number of ℓ_0-samplers used by our algorithm is bounded by $O(n^2)$, with high probability, none of them will fail. Hence we will not explicitly account for the probability of ℓ_0-samplers failure in our proofs.

DynamicStreamMatching stores two pairwise independent hash functions h_L and h_R to assign vertices to their groups, which requires $O(\log n)$ bits of space, the identities of all active pairs, which requires $\widetilde{O}(\gamma \cdot \beta)$ bits, and $O(\gamma \cdot \beta)$ ℓ_0-samplers for the active pairs during the stream, where each requires $O(\log^3 n)$ bits (Lemma 3.3.1). Hence, the total

space complexity of DynamicStreamMatching is:

$$\widetilde{O}(\gamma \cdot \beta) = \begin{cases} \widetilde{O}(n^2/\alpha^3) & \text{if } \alpha \leq \sqrt{n} \\ \widetilde{O}(n/\alpha) & \text{otherwise} \end{cases},$$

where we used the obvious bound of $\widetilde{\text{opt}} = O(n)$ and that $\beta = O(\log n)$ when $\alpha > \sqrt{n}$. Moreover, for any update on any edge (u, v), we apply h_L on u and h_R on v to identify the groups they belongs to, and update the ℓ_0-sampler for the edges between the groups $h_L(u)$ and $h_R(v)$ if they form an active pair. Therefore, the update time is $\text{polylog}(n)$.

We now prove the correctness of the algorithm. We begin by introducing some notation.

Notation. Fix a maximum matching M^* in G (of size opt). We say a vertex v is in M^* if v is matched by M^*. For any group $L_i \in \mathcal{L}$, (resp. $R_j \in \mathcal{R}$) each edge in M^* incident on L_i (resp. R_j) is referred to as a *matching edge* of this group. We say an (L_i, R_j) pair is *matchable* if L_i and R_j share at least one matching edge.

The general idea behind the proof is to treat each group as a single vertex (which forms a new graph $\mathcal{G}$), and to show that the ℓ_0-samplers we stored for $\mathcal{G}$ contain an $O(1)$-approximate matching for $\mathcal{G}$ which in turn leads to an $O(\alpha)$-approximate matching in G. More specifically, we show that there exists a subset of the groups in $\mathcal{L}$ and $\mathcal{R}$ where in the subgraph of $\mathcal{G}$ induced by this subset, each vertex has bounded degree while the total number of edges is sufficiently large. Then, using the following well-known fact (which we give a simple proof here for completeness), we can conclude that Algorithm DynamicStreamMatching outputs a large matching in $\mathcal{G}$, which will be an $O(\alpha)$-approximate matching in G.

Proposition 3.4.1. *Let $G(L, R, E)$ be a bipartite graph with m edges and max-degree d. Suppose for every vertex $u \in L$, we pick one edge incident on u uniformly at random; then w.p. at least $1 - \exp(-\Theta(m/d))$, the sampled edges contain a matching of size $\Omega(m/d)$.*

Proof. Follows from a simple balls and bins experiment in which we throw $N := |L|$ balls into $B := |R|$ bins. The number of non-empty bins in this process is equal to number of vertices in R which are "hit" by a vertex in L, which is clearly a lower bound on the size of a maximum matching in the sampled edges. Moreover, number of non-empty bins is $\Omega(\min\{N, B\})$ with probability $1 - \exp(-\Theta(\min\{N, B\}))$ by Proposition 2.1.5. Since maximum degree of G is d and G has m edges, $\min\{N, B\} \geq m/d$, finalizing the proof. □

We now present the proof of Theorem 3.6. We start by examining the number of edges of M^* that end up in different pairs of (L_i, R_j) groups. Since we only consider the edges in M^*, and the grouping leads to all edges between each (L_i, R_j) pair treated as a single edge, it is crucial that enough edges of M^* remain in distinct pair of groups.

Claim 3.4.2. *With probability at least* 0.5*, the number of edges of* M^* *that appear in different pairs of* (L_i, R_j) *groups is at least* $\min\{\text{opt}/32, \gamma^2/2\}$.

Proof. We will consider two cases. First suppose $\text{opt} > 4\gamma^2$. Let $Y_{i,j}$ be the random variable counting the number of edges in M^* that appear in (L_i, R_j). The total number of distinct (L_i, R_j) pairs is γ^2, and each edge in M^* appears in any (L_i, R_j) pair with probability $1/\gamma^2$. Hence $\mathbb{E}[Y_{i,j}] = \text{opt}/\gamma^2 > 4$. Since the end points of any two edges of M^* are independently assigned to the groups $\mathcal{L}$ and $\mathcal{R}$ (using pairwise independent hash functions h_L and h_R), $\mathbb{V}\text{ar}[Y_{i,j}] \leq \mathbb{E}[Y_{i,j}]$. By Chebyshev inequality (Proposition 2.1.1),

$$\Pr(Y_{i,j} = 0) \leq \Pr(|Y_{i,j} - \mathbb{E}[Y_{i,j}]| \geq \mathbb{E}[Y_{i,j}]) \leq \frac{\mathbb{V}\text{ar}[Y_{i,j}]}{(\mathbb{E}[Y_{i,j}])^2} \leq \frac{1}{\mathbb{E}[Y_{i,j}]} \leq \frac{1}{4}.$$

Hence, the expected number of (L_i, R_j) pairs that do not contain any edge from M^* is at most $\gamma^2/4$, and by Markov inequality, with probability at least 0.5, the number of (L_i, R_j) pairs that do not contain an edge from M^* is at most $\gamma^2/2$.

Now suppose $\text{opt} \leq 4\gamma^2$. Consider the first $\text{opt}/16$ edges of M^*. For any two edges e_1 and e_2 in M^*, the probability that e_1 and e_2 belong to the same (L_i, R_j) pair, for some L_i and R_j, (i.e., e_1 and e_j collide) is $1/\gamma^2$. Therefore, the expected number of collisions between the first $\text{opt}/16$ edges is $(\text{opt}/16)^2/\gamma^2 \leq \text{opt}/64$ (since $\text{opt} \leq 4\gamma^2$). Hence, with probability at least 0.5, the total number of collision is less than $\text{opt}/32$. Since all collisions can be resolved after removing $\text{opt}/32$ edges, at least $(\text{opt}/16 - \text{opt}/32) = \text{opt}/32$ edges of M^* are assigned to distinct (L_i, R_j) pairs. □

In the following, we focus on the case where at least $\min\{\text{opt}/32, \gamma^2/2\}$ edges of M^* appears in distinct (L_i, R_j) pairs. By Claim 3.4.2, this happens with probability at least 0.5. We consider the cases for $\gamma^2/2$ edges (Lemma 3.4.3) and $\text{opt}/32$ edges (Lemma 3.4.4) separately, and prove that in each case, the algorithm outputs an $O(\alpha)$-approximate matching, hence proving Theorem 3.6.

Lemma 3.4.3. *If at least* $\gamma^2/2$ *edges of* M^* *appear in distinct* (L_i, R_j) *pairs, then the algorithm* DynamicStreamMatching *outputs a matching of size* $\Omega(\text{opt}/\alpha)$ *w.p. at least* $1/4$.

Proof. If at least $\gamma^2/2$ edges of M^* appears in distinct (L_i, R_j) pairs, then at least $1/4$ fraction of the groups in $\mathcal{L}$ (denoted by $\mathcal{L}'$) have at least $\gamma/3$ different matchable groups in R_j. Otherwise, the total number of edges incident on $\mathcal{L}$ would be strictly less than $\gamma/4 \cdot \gamma + 3\gamma/4 \cdot (\gamma/3) = \gamma^2/2$, which is a contradiction. Then, the groups $\mathcal{L}'$ and $\mathcal{R}$ forms a graph (treating each group as a singe vertex) with at least $(\gamma/3) \cdot (\gamma/4)$ edges where each vertex has degree at most γ (there are only γ groups on each side).

It remains to show that any L_i in $\mathcal{L}'$ will pick at least one matchable $R_j \in \mathcal{R}$ as an active partner with probability $1 - 1/n^2$, and moreover, the matchable R_j is chosen uniformly at random. We can then apply Proposition 3.4.1 to complete the argument. Each L_i is matchable to $1/3$ fraction of the groups in $\mathcal{R}$ and since L_i picks more than $6 \log n$ active partners (independently and uniformly at random), by Chernoff bounds, L_i will pick a matchable R_j with probability at least $1-1/n^2$. By union bound, all L_i's in $\mathcal{L}'$ will pick at least one matchable $R_j \in \mathcal{R}$. Moreover, for each L_i in $\mathcal{L}'$, a matchable R_j would be picked uniformly at randomly from all groups matchable to L_i. Now, by Proposition 3.4.1, the edges returned by these matchable (L_i, R_j) pairs contain a matching of size $\Omega(\gamma^2/\gamma) = \Omega(\text{opt}/\alpha)$ with probability at least $(1 - \exp(-\Theta(\gamma))$. Since $\gamma = \left\lceil \widetilde{\text{opt}}/\alpha \right\rceil \geq 500$, the probability of failure is at most $1/2$, and the total probability that DynamicStreamMatching outputs a matching of size $\Omega(\text{opt}/\alpha)$ is at least $1/4$. □

Lemma 3.4.4. *If at least* $\text{opt}/32$ *edges of* M^* *appear in distinct* (L_i, R_j) *pairs, then the algorithm* DynamicStreamMatching *outputs a matching of size* $\Omega(\text{opt}/\alpha)$ *w.p. at least* 0.15.

Proof. We need some additional definition for this case. We say a group $L_i \in \mathcal{L}$ (resp. $R_j \in \mathcal{R}$) is *good* if the number of vertices in M^* that belong to L_i (resp. R_j) is in range $[0.999\alpha, 1.001\alpha]$. The rest of the groups are *bad*. We first show that most groups are good.

Claim 3.4.5. *W.p. at least* 0.9*, at most* 0.001 *fraction of the groups in* $\mathcal{L}$ *and* $\mathcal{R}$ *are bad.*

Proof. We only prove for the argument for the groups in $\mathcal{L}$; the same argument works for $\mathcal{R}$, as well. For each group $L_i \in \mathcal{L}$, let X^i be the random variable counting the number of vertices in M^* that are in L_i, we show that $\Pr\left(\left|X^i - \alpha\right| \geq 0.001\alpha\right) \leq 0.0001$. Then in expectation, at most 0.0001 fraction of the groups in $\mathcal{L}$ are bad, and by Markov inequality, w.p. at most $1/10$, more than 0.001 fraction of the groups are bad, proving the claim.

Let $L(M^*)$ be the set of vertices in L that are matched in M^*. For any vertex $u \in L(M^*)$, define X^i_u to be the indicator random variable denoting whether u belongs to L_i. We have $X^i = \sum_{u \in L(M^*)} X^i_u$. The expectation of X^i is

$$\mathbb{E}[X^i] = \sum_{u \in L(M^*)} \mathbb{E}[X^i_u] = |L(M^*)| \cdot (1/\gamma) = \text{opt} \cdot (\alpha/\text{opt}) = \alpha.$$

Since we use a pairwise independent hash function h_L to assign vertices in L to groups in $\mathcal{L}$, $\mathbb{V}\text{ar}\left[X^i\right] \leq \mathbb{E}\left[X^i\right]$. By Chebyshev inequality (Proposition 2.1.1), $\Pr\left(\left|X^i - \alpha\right| \geq 0.001\alpha\right) \leq \Pr\left(\left|X^i - \mathbb{E}[X^i]\right| \geq 100\sqrt{\alpha}\right) \leq 0.0001$, for sufficiently large n. □ Claim 3.4.5

Consider the joint event that (i) at least $\text{opt}/32$ edges of M^* appear in distinct (L_i, R_j) pairs, (ii) at most 0.0001 fraction of $\mathcal{L}$ are bad, and (iii) at most 0.0001 fraction of $\mathcal{R}$ are

bad. By Claim 3.4.5, this event happens with probability at least $1-1/2-1/10-1/10 = 0.3$. Moreover, the total number of edges in M^* that are incident on good groups in $\mathcal{L}$ is at least $0.999\alpha \cdot 0.999\gamma \geq 0.998\text{opt}$. Therefore, removing the bad groups in $\mathcal{L}$ only removes $0.002 \cdot \text{opt}$ edges of M^*. Similarly, removing the bad groups in $\mathcal{R}$ only removes $0.002 \cdot \text{opt}$ edges of M^*. Therefore, in the worst case, the total number of edges in M^* that appear in distinct (L_i, R_j) pairs where both L_i and R_j are good groups is at least $(1/32 - 0.002 \cdot 2) \cdot \text{opt} \geq \text{opt}/40$.

Now, opt/40 edges are incident on (at most) γ groups on each side where each group is only incident on at most 1.001α of them. Hence, at least $1/80$ fraction of the good $\mathcal{L}$ groups (denoted by $\mathcal{L}''$) must be incident on at least $\alpha/100$ edges, since otherwise, the total number of edges incident on $\mathcal{L}$ would be strictly less than $\gamma/80 \cdot 1.001\alpha + 79\gamma/80 \cdot \alpha/100 < \text{opt}/40$, a contradiction.

For each group L_i in $\mathcal{L}''$, $\alpha/100$ good $\mathcal{R}$ groups are matchable to L_i. Since L_i picks at least $\frac{100 \cdot \text{opt} \log n}{\alpha^2}$ active partners and each time, the picked active partner is matchable to L_i with probability at least $(\alpha/100)/\gamma = \alpha^2/(100 \cdot \text{opt})$, by Chernoff bounds, with high probability, we will pick at least one matchable group in $\mathcal{R}$ for L_i and the first picked matchable group is chosen uniformly at random from all matchable groups of L_i. By Proposition 3.4.1, DynamicStreamMatching outputs a matching of size $\Omega(\text{opt}/\alpha)$ with probability at least $(1 - \exp(-\Theta(\text{opt}/\alpha))$. Since $\text{opt} \geq 10^3\alpha$, the probability of failure is at most $1/2$, and hence the total probability of success is at least $0.3 \cdot 0.5 = 0.15$. □ Lemma 3.4.4

Theorem 3.6 now follows immediately from Claim 3.4.2 and Lemmas 3.4.3 and 3.4.4.

3.5. A Space Lower Bound for α-Approximation of Matching

In this section, we formalize Result 3.2 which shows that any dynamic streaming algorithm for approximating matchings requires $n^{2-o(1)}/\alpha^3$ space in order to achieve an α-approximation. By the connection between simultaneous communication complexity and dynamic streaming algorithms established by [230, 14] (Proposition 2.7.11), it suffices to prove a simultaneous communication lower bound for matching to obtain Result 3.2.

We define $\mathsf{ApxMatch}_{n,k,\alpha}$ as the k-party communication problem of α-approximating matching in an n-vertex input graph $G(V, E)$, partitioned across k players. We have,

Theorem 3.7. *For any $\alpha \leq \sqrt{n}$, there exists some $k = \alpha \cdot \left(\frac{n}{\alpha}\right)^{o(1)}$ such that the k-party simultaneous communication complexity of finding an α-approximate matchings is*

$$\mathsf{R}^{||}_{1/10}(\mathsf{ApxMatch}_{n,k,\alpha}) = k \cdot \Omega(n^2/\alpha^3).$$

We remark that although we state Theorem 3.7 for general graphs, using the same reduction based on random partitioning of vertices mentioned earlier in Section 3.4, the

same lower bound also holds for bipartite graphs.

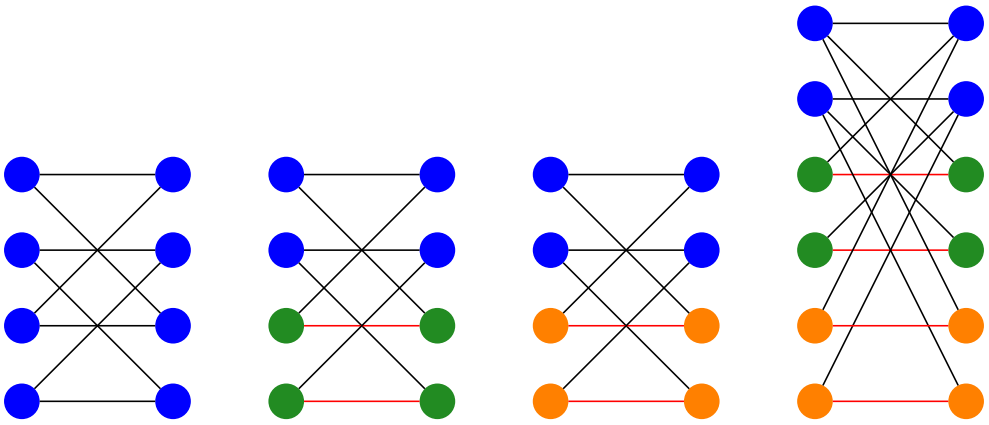
By the easy direction of Yao's minimax principle (Proposition 2.7.2), to prove Theorem 3.7, it suffices to prove the lower bound for deterministic protocols over a fixed distribution $\mathcal{D}$ of the inputs. In our hard distribution, each player will be given an (r,t)-RS graph with half of the edges discarded uniformly at random from each induced matchings (see Section 2.3 for a reminder on definition of RS graphs). The final graph is constructed in a *correlated* way where for each player, only one of the induced matchings is "special" and the all other edges will be incident on the same set of vertices. We carefully choose the parameters such that the referee/coordinator has to know the edges of the private induced matchings for outputting a large matching. However, since each player is unaware of the identity of his special matching, he has to send enough information for recovering a large fraction of the edges from *every* induced matching.

A hard input distribution $\mathcal{D}$. (parameterized by a sufficiently large integer $N > 0$)

Parameters: $r = N^{1-o(1)}$, $t = \frac{\binom{N}{2}-o(N^2)}{r}$, $k = \frac{10\alpha \cdot N}{r}$, $n = N + 2 \cdot k \cdot r$.

1. Fix an (r,t)-RS graph G^{RS} on N vertices with induced matchings $M_1^{\mathsf{RS}}, \ldots, M_t^{\mathsf{RS}}$.
2. Pick $j^* \in [t]$ uniformly at random.
3. For each player $P^{(i)}$ independently,
   (a) Denote by G_i the input graph of $P^{(i)}$, initialized to be a copy of G^{RS} with vertices $V_i = [N]$. Moreover, define V_i^* as the set of vertices incident on $M_{j^*}^{\mathsf{RS}}$.
   (b) Drop each edge in G_i w.p. $1/2$ independently and keep the remaining edges.
4. Pick a random permutation σ of $[n]$. For every player $P^{(i)}$, for each vertex v in $V_i \setminus V_i^*$ with label ℓ, *relabel* v with $\sigma(\ell)$. Enumerate the vertices in V_i^* (from the one with the smallest label to the largest), and relabel the ℓ-th enumerated vertex with $\sigma(N + (i-1) \cdot 2r + \ell)$. In the final graph, the vertices with the same label correspond to the same vertex.

In the input graph G_i of player $P^{(i)}$ in $\mathcal{D}$, those vertices whose labels belong to $\sigma([N])$ are referred to as *shared* vertices since they belong to the input graph of every player, and the vertices V_i^* are referred to as the *private* vertices as they only appear in the input graph of $P^{(i)}$ (in the final graph, i.e., after relabeling). We refer to the induced matching between private vertices of any player $P^{(i)}$ as the *special* matching of $P^{(i)}$. We point out that, in general, the final graph constructed by this distribution is a multi-graph with n vertices and $O(kN^2) = O(n^2/\alpha)$ edges (counting the multiplicities); the multiplicity of each edge is also at most k. Finally, the existence of an (r,t)-RS graph G^{RS} with the parameters used



(a) Local view of a single player. (b) Special matching of player one. (c) Special matching of player two. (d) Global view.

Figure 3: Illustration of the hard distribution $\mathcal{D}$ in Theorem 3.7 with two players.

in this distribution is guaranteed by a result of [18] (see Proposition 2.3.1).

We prove that distributional complexity of ApxMatch on $\mathcal{D}$ is "large", i.e.,

$$\mathsf{D}^{||}_{\mathcal{D},1/3}(\mathsf{ApxMatch}_{n,k,\alpha}) \geq k \cdot rt/10\alpha = k \cdot \Omega(n^2/\alpha^3). \tag{3.1}$$

We remark that our proof of this equation in this thesis differs significantly from our earlier proof published in [35] (see also [34]), and follows more closely the framework we will introduce in Chapter 8 for proving multi-party communication complexity lower bounds. This further allows us to strenghten the bounds in [35] by an $n^{o(1)}$ factor.

Notation. In the following, fix any deterministic (1/3)-error protocol π for ApxMatch. Let us use Σ and J to denote the random variable for permutation σ and index $j^* \in [t]$ in $\mathcal{D}$. We further define $\mathsf{M}_{i,j}$ for any $i \in [k]$ and $j \in [t]$ to denote the random variable for the matching M_j^{RS} (after dropping some of its edges) in graph G_i of player $P^{(i)}$. Finally, recall that $\Pi := (\Pi_1, \ldots, \Pi_k)$ denotes the transcript of the messages sent from players $P^{(1)}, \ldots, P^{(k)}$ to the referee in π. We assume that the referee (but definitely not the players) is additionally given the permutation σ and index j^*; this clearly can only strengthen our lower bound proof. This means that the output matching of referee is a function of Π as well as Σ, J. We denote this matching by $M_\pi(\Pi, \Sigma, \mathsf{J})$.

In distribution $\mathcal{D}$, all edges except for the special matchings $\mathsf{M}_{1,\mathsf{J}}, \ldots, \mathsf{M}_{k,\mathsf{J}}$ are incident on the same set of shared vertices. As such, $M_\pi(\Pi, \Sigma, \mathsf{J})$ needs to have a "large" intersection with $\mathsf{M}_{1,\mathsf{J}}, \ldots, \mathsf{M}_{k,\mathsf{J}}$ to be a large matching. But for this to happen, the players need to make sure that the "uncertainty" about $\mathsf{M}_{1,\mathsf{J}}, \ldots, \mathsf{M}_{k,\mathsf{J}}$ is "low" for the referee. At the same time,

the players are oblivious to which matching in their input is $\mathsf{M}_{i,\mathsf{J}}$, i.e. is special, and which ones are not special. Consequently, to reduce the uncertainty about $\mathsf{M}_{i,\mathsf{J}}$, the player $P^{(i)}$ needs to reduce the uncertainty about *all* induced matchings in his input. We use this intuition to lower bound the communication cost of π.

We start by bounding the size of the matching output by the referee based on the information revealed about special matchings by the protocol.

Lemma 3.5.1. $\mathbb{E}\left|M_\pi(\Pi,\Sigma,\mathsf{J})\right| \leq N + \mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\Pi \mid \Sigma,\mathsf{J})$.

Proof. All edges in G except for $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$ are incident on at most N shared vertices. Hence, the total contribution of all those edges to $M_\pi(\Pi,\Sigma,\mathsf{J})$ is at most N. All the remaining edges in $M_\pi(\Pi,\Sigma,\mathsf{J})$, denoted by $M'_\pi(\Pi,\Sigma,\mathsf{J})$, now belong to $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$.

Fix any assignment of Π,σ,j^* for (Π,Σ,J). Given Π,σ,j^*, the referee is only able to output an edge e in the final matching $M'_\pi(\Pi,\sigma,j^*)$ (note that this matching is now fixed), iff e always appear in $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Pi=\Pi,\Sigma=\sigma,\mathsf{J}=j^*$, as otherwise the referee may output an edge which does not belong to the input graph, a contradiction.

Let $\ell := |M'_\pi(\Pi,\sigma,j^*)|$. The above discussion implies that at least ℓ edges in $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$ are "fixed" conditioned on $\Pi=\Pi$, $\Sigma=\sigma$ and $\mathsf{J}=j^*$. This implies that:

$$\mathbb{H}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Pi=\Pi,\Sigma=\sigma,\mathsf{J}=j^*) \leq rk-\ell,$$

as the support of $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$ has size $2^{rk-\ell}$ (after fixing ℓ edges to always be present) and by Fact 2.6.1-(1). By taking expectation over all Π,σ,j^*, we have,

$$\begin{aligned}\mathbb{H}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Pi,\Sigma,\mathsf{J}) &= \mathbb{E}\left[\mathbb{H}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Pi=\Pi,\Sigma=\sigma,\mathsf{J}=j^*)\right]\\ &\leq rk - \mathbb{E}\left|M'_\pi(\Pi,\Sigma,\mathsf{J})\right| \leq rk - \left(\mathbb{E}\left|M_\pi(\Pi,\Sigma,\mathsf{J})\right| - N\right).\end{aligned}$$

Finally, using the fact that $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$ is uniform over its support (which has size 2^{rk}) conditioned on Σ,J, we have (by Fact 2.6.1-(1)),

$$\begin{aligned}\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\Pi \mid \Sigma,\mathsf{J}) &= \mathbb{H}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Sigma,\mathsf{J}) - \mathbb{H}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}} \mid \Pi,\Sigma,\mathsf{J})\\ &\geq \mathbb{E}\left|M_\pi(\Pi,\Sigma,\mathsf{J})\right| - N.\end{aligned}$$

This concludes the proof of the lemma. □

Our goal is now to bound $\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\Pi \mid \Sigma,\mathsf{J}) \leq N$ which would be sufficient to prove Eq (3.1) using Lemma 3.5.1. We first bound the information revealed about the matchings $\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}$ to the referee by the summation of information revealed by each

player $P^{(i)}$ about his special matching $\mathsf{M}_{i,\mathsf{J}}$.

Lemma 3.5.2. $\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}\mid\mathsf{\Sigma},\mathsf{J})\leq\sum_{i=1}^{k}\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J})$.

Proof. Intuitively, the lemma is true because after conditioning on $\mathsf{\Sigma}$ and J, the input of players become independent of each other (it would only be a function of which edges are dropped from G^{RS} given to each player). As a result, the messages communicated by one player do not give extra information about the special matching of another player.

Recall that $\mathsf{\Pi}=(\mathsf{\Pi}_1,\ldots,\mathsf{\Pi}_k)$. By chain rule of mutual information,

$$\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}_1,\ldots,\mathsf{\Pi}_k\mid\mathsf{\Sigma},\mathsf{J})\underset{\text{Fact 2.6.1-(6)}}{=}\sum_{i=1}^{k}\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Pi}^{<i},\mathsf{\Sigma},\mathsf{J}).$$

We first show that for each $i\in[k]$,

$$\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Pi}^{<i},\mathsf{\Sigma},\mathsf{J})\leq\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J}),\tag{3.2}$$

i.e., "dropping" the conditioning on $\mathsf{\Pi}^{<i}$ only increases the information. This is because $\mathsf{\Pi}_i\perp\mathsf{\Pi}^{<i}\mid\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}},\mathsf{\Sigma},\mathsf{J}$ as after conditioning on $\mathsf{\Sigma}$ and J, the inputs the players, and so their deterministic functions $\mathsf{\Pi}_i$ and $\mathsf{\Pi}^{<i}$ become independent of each other. Hence, we can apply Proposition 2.6.4. By chain rule of mutual information (Fact 2.6.1-(6)),

$$\mathbb{I}(\mathsf{M}_{1,\mathsf{J}},\ldots,\mathsf{M}_{k,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J})=\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J})+\mathbb{I}(\mathsf{M}_{-i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{M}_{i,\mathsf{J}},\mathsf{\Sigma},\mathsf{J})=\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J}),$$

since $\mathbb{I}(\mathsf{M}_{-i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{M}_{i,\mathsf{J}},\mathsf{\Sigma},\mathsf{J})=0$ as $\mathsf{\Pi}_i$ is independent of $\mathsf{M}_{-i,\mathsf{J}}$ after conditioning on $\mathsf{\Sigma},\mathsf{J}$. The lemma now follows from Eq (3.2) and above equation. □

We now prove that no player cannot reveal much information about his special matching without making a much larger communication (by a factor of t, i.e., the number of induced matchings in G^{RS}). This is established via a direct sum style argument (see Section 2.8.2) which argues that since the player is unaware of the identity of his special matching, he needs to reveal enough information on *every* matching he has in order to reveal enough information on the (unknown) special one.

Lemma 3.5.3. *For any* $i\in[k]$, $\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\mathsf{\Pi}_i\mid\mathsf{\Sigma},\mathsf{J})\leq|\Pi_i|/t$.

Proof. Define $\mathsf{\Phi}_i$ as (random variable for) the labeling function that labels the vertices of graph G_i given to player $P^{(i)}$. Function $\mathsf{\Phi}_i$ is uniquely determined by permutation $\mathsf{\Sigma}$ and index J. Additionally, the input to player $P^{(i)}$ (and consequently the message $\mathsf{\Pi}_i$) is uniquely determined by the matchings $\mathsf{M}_{i,1},\ldots,\mathsf{M}_{i,t}$ and the labeling function $\mathsf{\Phi}_i$ as they

fully determine the graph G_i. We first argue that, $\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\Pi_i \mid \Sigma, \mathsf{J}) \leq \mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\Pi_i \mid \Phi_i, \mathsf{J})$. As stated above, $\Pi_i \perp \Sigma \mid \mathsf{M}_{i,\mathsf{J}}, \Phi_i, \mathsf{J}$. As such, since Σ and J determine Φ_i, conditioning on whatever left of Σ beside Φ_i can only decrease the mutual information by Proposition 2.6.4, leading to this equation. We can bound the RHS in this equation as follows,

$$\begin{aligned}\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\Pi_i \mid \Phi_i, \mathsf{J}) &= \mathop{\mathbb{E}}_{j\sim\mathsf{J}}\left[\mathbb{I}(\mathsf{M}_{i,j}\,;\Pi_i \mid \Phi_i, \mathsf{J}=j)\right] \\ &= \frac{1}{t}\cdot\sum_{j=1}^{t}\mathbb{I}(\mathsf{M}_{i,j}\,;\Pi_i \mid \Phi_i, \mathsf{J}=j) = \frac{1}{t}\cdot\sum_{j=1}^{t}\mathbb{I}(\mathsf{M}_{i,j}\,;\Pi_i \mid \Phi_i),\end{aligned}$$

where the final equality is because the joint distribution of $(\mathsf{M}_{i,j}, \Pi_i, \Phi_i)$ is independent of the event $\mathsf{J}=j$. Finally,

$$\begin{aligned}\mathbb{I}(\mathsf{M}_{i,\mathsf{J}}\,;\Pi_i \mid \Phi_i, \mathsf{J}) = \frac{1}{t}\cdot\sum_{j=1}^{t}\mathbb{I}(\mathsf{M}_{i,j}\,;\Pi_i \mid \Phi_i) &\leq \frac{1}{t}\cdot\sum_{j=1}^{t}\mathbb{I}(\mathsf{M}_{i,j}\,;\Pi_i \mid \Phi_i, \mathsf{M}_i^{>j}) \\ &\qquad \text{(as } \mathsf{M}_i^{>j} \perp \mathsf{M}_{i,j} \text{ and Proposition 2.6.3)} \\ = \frac{1}{t}\cdot\mathbb{I}(\mathsf{M}_{i,1},\ldots,\mathsf{M}_{i,t}\,;\Pi_i \mid \Phi_i) \leq \frac{1}{t}\mathbb{H}(\Pi_i) &\leq \frac{1}{t}\cdot|\Pi_i|\,, \\ &\qquad \text{(by chain rule of mutual information (Fact 2.6.1-(6)) and Fact 2.6.1-(1))}\end{aligned}$$

finalizing the proof. □

Proof of Theorem 3.7. By the easy direction of Yao's minimax principle (Proposition 2.7.2), to prove Theorem 3.7, it suffices to prove Eq (3.1). Let π be any deterministic protocol on distribution $\mathcal{D}$ with communication cost $\|\pi\| < k\cdot rt/10\alpha$. By Lemmas 3.5.1, 3.5.2, and 3.5.3, we have that, expected size of the matching M_π output in this protocol is:

$$\mathbb{E}\,|M_\pi| \leq N + \frac{1}{t}\cdot\sum_{i=1}^{k}|\Pi_i| = N + \|\pi\|/t < N + k\cdot r/10\alpha \leq 2N. \qquad (\text{as } kr = 10\alpha N)$$

By Markov bound, this implies that with probability at least $1/2$, $|M_\pi| \leq 4N$. At the same time, there exists a matching of size $k\cdot r/2 - O(\sqrt{k\cdot r}) > 4\alpha N$ with high probability in $G\sim\mathcal{D}$ by simply taking all special matchings. This means that with probability at least $1/3$, the returned matching is not an α-approximate matching, a contradiction. □

3.6. Space Lower Bounds for α-Estimating Matching Size

We present our space lower bounds for α-estimation algorithms in dynamic streams. Similar to before, we prove this lower bounds in the simultaneous communication model and then invoke the results of [14, 230] (see Proposition 2.7.11) to extend them to dynamic streams.

3.6.1. An $\Omega(\sqrt{n}/\alpha^{2.5})$ Lower Bound for Sparse Graphs

We consider the sparse graphs case in this section (i.e., Part (1) of Result 3.3), and show that any single-pass streaming algorithm that computes an α-estimation of matching size must use $\Omega(\sqrt{n}/\alpha^{2.5})$ bits of space even if the input graph only has $O(n)$ edges.

Define the *sparse matching size estimation* problem, $\mathsf{SMS}_{n,k}$, as the following k-party communication problem: each player $P^{(i)}$ is given a matching M_i over a set V of $n + \frac{n}{k}$ vertices[2] and the goal of the players is to approximate the maximum matching size of $G(V, \bigcup_{i\in[k]} M_i)$ to within a factor smaller than $\frac{k+1}{2}$. We prove the following lower bound on the communication complexity of $\mathsf{SMS}_{n,k}$.

Theorem 3.8. *For integers $n, k \geq 2$, and any constant $\delta < 1/2$:* $\mathsf{R}^{\shortparallel}_{\delta}(\mathsf{SMS}_{n,k}) = \Omega\left(\frac{\sqrt{n}}{k\sqrt{k}}\right)$.

Part (1) of Result 3.3 immediately follows from Theorem 3.8 as we show below.

Proof of Result 3.3, Part (1). Any simultaneous protocol for estimating matching size to within a factor of $\alpha < \frac{k+1}{2}$ can be used to solve the $\mathsf{SMS}_{n,k}$ problem. As by Proposition 2.7.11, simultaneous communication complexity of a k-player problem is at most k times the space complexity of any single-pass streaming algorithm in dynamic streams; this finalizes the first part of the proof.

To see that the space complexity holds even when the input graph is both sparse and having bounded arboricity, notice that any graph G in $\mathsf{SMS}_{n,k}$ has exactly $k \cdot \frac{n}{k} = n$ edges (hence sparse); furthermore, since each player is given a matching (which is always a forest), the arboricity of G is at most $k \leq 2\alpha$. □

In the following, we focus on proving Theorem 3.8. This theorem is ultimately proved by a reduction from the BHM^0 problem defined in Section 3.3.1. However, this reduction is non-standard in the sense that it is *protocol-dependent*: given any protocol π for SMS, we create a protocol for BHM^0 by *embedding* an instance of BHM^0 in the input of SMS, whereby the embedding is designed specifically for the protocol π. It is worth mentioning that BHM^0 is a hard problem even in the one-way model, while the distribution that we create for SMS is only hard in the simultaneous communication model, meaning that if any player is allowed to send a single message to any other player (instead of the referee), then $O(\log n)$ bits of communication suffices to solve the problem. Therefore, a key technical challenge here is to design a reduction from a one-way problem to a problem that is "inherently" simultaneous.

[2] To simplify the exposition, we use $n + \frac{n}{k}$ instead of the usual n as the number of vertices.

A Hard Input Distribution for $\mathsf{SMS}_{n,k}$

Let $\mathcal{D}_{\mathsf{BHM}}$ be the hard input distribution of $\mathsf{BHM}^0_{\frac{2n}{k}}$ in Corollary 3.5 (for $t = 2$) and $\mathcal{D}^{\mathsf{Y}}_{\mathsf{BHM}}$ and $\mathcal{D}^{\mathsf{N}}_{\mathsf{BHM}}$ be, respectively, the distribution on Yes and No instances of $\mathcal{D}_{\mathsf{BHM}}$.

The distribution $\mathcal{D}_{\mathsf{SMS}}$ for $\mathsf{SMS}_{n,k}$:

1. For each $i \in [k]$, independently draw a $\mathsf{BHM}^0_{\frac{2n}{k}}$ instance $(M^{\mathsf{B}}_i, x^{\mathsf{B}}_i) \sim \mathcal{D}_{\mathsf{BHM}}$.
2. Draw a *random* permutation σ on $\left[n + \frac{n}{k}\right]$.
3. For each player $i \in [k]$, we define a mapping $\sigma_i : [\frac{2n}{k}] \to \left[n + \frac{n}{k}\right]$ as follows:
   - For each $j \in [\frac{2n}{k}]$ with $x^{\mathsf{B}}_i(j) = 1$, if $x^{\mathsf{B}}_i(j)$ is the ℓ-th smallest index with value 1, let $\sigma_i(j) := \sigma(\ell)$[a].
   - For each $j \in [\frac{2n}{k}]$ with $x^{\mathsf{B}}_i(j) = 0$, if $x^{\mathsf{B}}_i(j)$ is the ℓ-th smallest index with value 0, let $\sigma_i(j) := \sigma(i \cdot \frac{n}{k} + \ell)$.
4. The input to each player $P^{(i)}$ is a matching $M_i := \left\{(\sigma_i(u), \sigma_i(v)) \mid (u, v) \in M^{\mathsf{B}}_i\right\}$.

[a] Here, we use the fact that $\|x^{\mathsf{B}}_i\|_0 = \frac{n}{k}$ in $\mathcal{D}_{\mathsf{BHM}}$ by Corollary 3.5

Observe that the distribution $\mathcal{D}_{\mathsf{SMS}}$ is defined by k instances of $\mathsf{BHM}^0_{\frac{2n}{k}}$, i.e., $(M^{\mathsf{B}}_i, x^{\mathsf{B}}_i)$ (for $i \in [k]$), along with a mapping σ. The mapping σ relates the vectors x^{B}_i to the set of vertices in the final graph G while ensuring that across the players, for any $j \in [\frac{2n}{k}]$ where $x^{\mathsf{B}}_i(j) = 1$, the vertex that j maps to is *shared*, while the vertices with $x^{\mathsf{B}}_i(j) = 0$ are *unique* to each player. Moreover, the mapping σ_i provided to each player effectively describes the set of vertices (denoted by V_i) that the edges of $P^{(i)}$ will be incident on, and the matching M^{B}_i describes the edges between V_i. Hence, we can *uniquely* define the input of each player $P^{(i)}$ by the pair $(M^{\mathsf{B}}_i, \sigma_i)$, and from now on, without loss of generality, we assume the input given to each player $P^{(i)}$ is the pair $(M^{\mathsf{B}}_i, \sigma_i)$.

We should note right away that the distribution $\mathcal{D}_{\mathsf{SMS}}$ is not a "hard" distribution for $\mathsf{SMS}_{n,k}$ in the traditional (distributional) sense: it is not hard to verify that for any graph $G \sim \mathcal{D}_{\mathsf{SMS}}$, maximum matching size $\mathsf{MM}(G)$ of G is concentrated around its expectation, and hence it is trivial to design a protocol when instances are promised to be *only* sampled from $\mathcal{D}_{\mathsf{SMS}}$: always output $\mathbb{E}_{G \sim \mathcal{D}_{\mathsf{SMS}}}[\mathsf{MM}(G)]$, which needs no communication.

Nevertheless, the way we use the distribution $\mathcal{D}_{\mathsf{SMS}}$ as a hard distribution is to consider any protocol π_{SMS} that succeeds *uniformly*, i.e., on *any* instance of $\mathsf{SMS}_{n,k}$; we then execute π_{SMS} on $\mathcal{D}_{\mathsf{SMS}}$ and argue that in order to perform well on every instance of $\mathcal{D}_{\mathsf{SMS}}$, π_{SMS} must convey a non-trivial amount of information about the input of the players in *some sub-distribution* of $\mathcal{D}_{\mathsf{SMS}}$. To continue, we need the following definitions.

Definition 3.2 (Input Profile)**.** *For each graph $G \sim \mathcal{D}_{\mathsf{SMS}}$, we define the input profile of G*

to be a vector $f \in \{\mathsf{Yes}, \mathsf{No}\}^k$*, where* $f(i) = \mathsf{Yes}$ *iff the* i*-th* BHM *instance* $(M_i^{\mathsf{B}}, x_i^{\mathsf{B}})$ *in* G *is a* Yes *instance and otherwise* $f(i) = \mathsf{No}$*.*

The 2^k different possible input profiles partition $\mathcal{D}_{\mathsf{SMS}}$ into 2^k different distributions. For any input profile f, we use $\mathcal{D}_{\mathsf{SMS}} \mid f$ to denote the distribution of $\mathcal{D}_{\mathsf{SMS}}$ *conditioned* on its input profile being f. Two interesting profiles for our purpose are the *all-equal* profiles, i.e., $f_{\mathsf{Yes}} := (\mathsf{Yes}, \ldots, \mathsf{Yes})$ and $f_{\mathsf{No}} := (\mathsf{No}, \ldots, \mathsf{No})$, due to the following claim.

Claim 3.6.1. *For any graph* $G \sim (\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$*,* $\mathsf{MM}(G) \geq \frac{n}{2} + \frac{n}{2k}$*, while for any graph* $G \sim (\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$*,* $\mathsf{MM}(G) \leq \frac{n}{k}$*.*

Proof. In $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$, each BHM instance $(M_i^{\mathsf{B}}, x_i^{\mathsf{B}})$ (for $i \in [k]$) is drawn from $\mathcal{D}_{\mathsf{SMS}}^{\mathsf{Y}}$, meaning that for every edge $(u, v) \in M_i^{\mathsf{B}}$, $x_i^{\mathsf{B}}(u) \oplus x_i^{\mathsf{B}}(v) = 0$. Therefore, either $x_i^{\mathsf{B}}(u) = x_i^{\mathsf{B}}(v) = 0$ or $x_i^{\mathsf{B}}(u) = x_i^{\mathsf{B}}(v) = 1$. Since M_i^{B} is a perfect matching over the set $[\frac{2n}{k}]$ and the hamming weight of x_i^{B} is $\frac{n}{k}$ (by Corollary 3.5), for half of the edges in M_i^{B}, we must have $x_i^{\mathsf{B}}(u) = x_i^{\mathsf{B}}(v) = 0$. Moreover, as $\mathcal{D}_{\mathsf{SMS}}$ maps every vertex with $x_i^{\mathsf{B}}(j) = 0$ to a distinct vertex in G, these $\frac{1}{2} \cdot \left|M_i^{\mathsf{B}}\right| = \frac{n}{2k}$ edges are vertex-disjoint with any other edge in the final graph G. Hence, between the k players, these edges together form a matching of size $k \cdot \frac{n}{2k} = \frac{n}{2}$. Finally, there is also a matching of size $\frac{n}{2k}$ between the shared vertices: simply use the edges corresponding to a matching M_i^{B} of an arbitrary player $P^{(i)}$ that are incident on shared vertices. This means that in this case, $\mathsf{MM}(G) \geq \frac{n}{2} + \frac{n}{2k}$.

In $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$, each BHM instance $(M_i^{\mathsf{B}}, x_i^{\mathsf{B}})$ (for $i \in [k]$) is drawn from $\mathcal{D}_{\mathsf{SMS}}^{\mathsf{N}}$, meaning that for every edge $(u, v) \in M_i^{\mathsf{B}}$, $x_i^{\mathsf{B}}(u) \oplus x_i^{\mathsf{B}}(v) = 1$. Therefore, exactly one of $x_i^{\mathsf{B}}(u)$ or $x_i^{\mathsf{B}}(v)$ is equal to 1. In $\mathcal{D}_{\mathsf{SMS}}$, for every player, the vertices where $x_i^{\mathsf{B}}(j) = 1$ are all mapped to the (same) set of vertices $\left\{\sigma(1), \sigma(2), \ldots, \sigma(\frac{n}{k})\right\}$ (denoted by V_0). Therefore, in the final graph G, *every* edge of *every* player is incident on some vertex in V_0, and hence the maximum matching size in G is at most $|V_0| = \frac{n}{k}$. □

In the following, we fix any δ-error protocol π_{SMS} for $\mathsf{SMS}_{n,k}$. By Claim 3.6.1, π_{SMS} is also a δ-error protocol for distinguishing between the two distributions $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ and $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$: simply output Yes if the estimate of $\mathsf{MM}(G)$ is strictly larger than $\frac{n}{k}$ and output No otherwise. From here on, with a slight abuse of notation, we say that π_{SMS} outputs Yes whenever it estimates $\mathsf{MM}(G)$ strictly larger than $\frac{n}{k}$ and outputs No otherwise (this is defined for *any* input, not necessarily chosen from $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ or $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$).

Intuitively, to distinguish between $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ and $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$, one should solve (at least one of) the BHM^0 instances embedded in the distribution. This naturally suggests the possibility of performing a reduction from BHM^0 and arguing that the distribution on $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ and $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$ is a hard distribution for $\mathsf{SMS}_{n,k}$. However, in the case of

these two distributions, the k BHM^0 instances are highly correlated and hence it is hard to reason about which BHM^0 instance is "actually being solved". To get around this, we try π_{SMS} on other input profiles, with, informally speaking, less correlation across the BHM instances. An immediate issue here is that, unlike the case for the distributions $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ and $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$, the matching sizes for graphs drawn from the other input profiles do not have a large gap. Hence, a priori it is not even clear what the actual task of π_{SMS} is, or why π_{SMS} should be able to distinguish them. However, we show that there are special pairs of input profiles (other than f_{Yes} and f_{No}) with our desired property (i.e., "low" correlation between the BHM^0 instances) that π_{SMS} is still able to distinguish. These pairs are ultimately connected to the (property of) protocol π_{SMS} itself and hence vary across different choices for the protocol π_{SMS}; this is the main reason that we perform a protocol-dependent reduction in our proof.

For any input profile f, define p_f^{Y} (resp. p_f^{N}) as the probability that π_{SMS} outputs Yes (resp. No) when its input is sampled from $\mathcal{D}_{\mathsf{SMS}} \mid f$. We define the notation of *informative index* for the protocol π_{SMS}.

Definition 3.3 (Informative Index)**.** *We say that an index $i \in [k]$ is* γ-informative *for the protocol π_{SMS} iff there exist two input profiles f and g where $f(i) = \mathsf{Yes}$, $g(i) = \mathsf{No}$, and $f(j) = g(j)$ for all $j \neq i$, such that $p_f^{\mathsf{Y}} + p_g^{\mathsf{N}} \geq 1 + 2\gamma$. In this case, the input profiles f and g are called the* witness *of i.*

Informally speaking, if π_{SMS} has a γ-informative index i, then π_{SMS} can distinguish whether the i-th BHM^0 instance is a Yes or No instance w.p. at least $\frac{1}{2} + \gamma$ (i.e., π_{SMS} solves the i-th BHM^0 instance). In the rest of this section, we prove that indeed every protocol π_{SMS} has an informative index.

Lemma 3.6.2. *Any δ-error protocol π_{SMS} for SMS has a γ-informative index for $\gamma = \frac{1-2\delta}{2k}$.*

Proof. Suppose towards a contradiction that for any two input profiles f and g that differ only on one entry (say i, and $f(i) = \mathsf{Yes}$, $g(i) = \mathsf{No}$), we have, $p_f^{\mathsf{Y}} + p_g^{\mathsf{N}} < 1 + 2\gamma$ for $\gamma = \frac{1-2\delta}{2k}$.

Consider the following sequence of $(k+1)$ input profiles:

$$(f_{\mathsf{Yes}} =)(\mathsf{Yes}, \mathsf{Yes}, \ldots, \mathsf{Yes}), (\mathsf{No}, \mathsf{Yes}, \ldots, \mathsf{Yes}), (\mathsf{No}, \mathsf{No}, \ldots, \mathsf{Yes}), \ldots, (\mathsf{No}, \mathsf{No}, \ldots, \mathsf{No})(= f_{\mathsf{No}})$$

whereby, for the j-th input profile of this sequence (denoted by f_j), the first $j-1$ entries of f_j are all No, and the rest are all Yes.

Observe that for any $j \in [k]$, the input profiles f_j and f_{j+1} differ in exactly one entry j, and $f_j(j) = \mathsf{Yes}$, while $f_{j+1}(j) = \mathsf{No}$. Hence, by our assumption, we have $p_{f_j}^{\mathsf{Y}} + p_{f_{j+1}}^{\mathsf{N}} <$

$1 + 2\gamma$, which implies $p^{\mathsf{Y}}_{f_j} < 1 + 2\gamma - p^{\mathsf{N}}_{f_{j+1}} = p^{\mathsf{Y}}_{f_{j+1}} + 2\gamma$ as $p^{\mathsf{Y}}_{f_{j+1}} + p^{\mathsf{N}}_{f_{j+1}} = 1$. Therefore, $p^{\mathsf{Y}}_{f_1} < p^{\mathsf{Y}}_{f_2} + 2\gamma < p^{\mathsf{Y}}_{f_3} + 2\gamma \cdot 2 < \cdots < p^{\mathsf{Y}}_{f_{k+1}} + 2\gamma \cdot k$, which implies (by adding $p^{\mathsf{N}}_{f_{k+1}}$ to both sides of the inequality)

$$p^{\mathsf{Y}}_{f_1} + p^{\mathsf{N}}_{f_{k+1}} < p^{\mathsf{Y}}_{f_{k+1}} + p^{\mathsf{N}}_{f_{k+1}} + 2\gamma \cdot k = 1 + 2\gamma \cdot k = 2 \cdot (1 - \delta), \tag{3.3}$$

by our choice of γ. However, since π_{SMS} is a δ-error protocol for $\mathsf{SMS}_{n,k}$, by Claim 3.6.1, the probability that π_{SMS} succeeds in distinguishing $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}})$ from $(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$ on the distribution $\frac{1}{2}(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{Yes}}) + \frac{1}{2}(\mathcal{D}_{\mathsf{SMS}} \mid f_{\mathsf{No}})$ is at least $1 - \delta$. Therefore, $\frac{1}{2} \cdot (p^{\mathsf{Y}}_{f_1} + p^{\mathsf{N}}_{f_{k+1}}) \geq 1 - \delta$, a contradiction to Eq (3.3). □

In the next section, we use existence of a γ-informative index in any protocol π_{SMS} for $\mathsf{SMS}_{n,k}$ to obtain a protocol for $\mathsf{BHM}^0_{\frac{2n}{k}}$ w.p. of success at least $\frac{1}{2} + \gamma$, based π_{SMS}.

The Reduction From the $\mathsf{BHM}^0_{\frac{2n}{k}}$ Problem

Recall that π_{SMS} is a δ-error protocol for the distribution $\mathcal{D}_{\mathsf{SMS}}$. Let i^* be a γ-informative index of π_{SMS} (as in Lemma 3.6.2), and let input profiles f_{i^*} and g_{i^*} be the witness of i^*. We design the following protocol π_{BHM} using π_{SMS} as a sub-routine.

Protocol π_{BHM}. A protocol for reducing $\mathsf{BHM}^0_{\frac{2n}{k}}$ to $\mathsf{SMS}_{n,k}$

Input: An instance $(M, x) \sim \mathcal{D}_{\mathsf{BHM}}$ of $\mathsf{BHM}^0_{\frac{2n}{k}}$.
Output: Yes if $Mx = 0^{\frac{n}{k}}$ and No if $Mx = 1^{\frac{n}{k}}$.

---

1. Bob creates the input $(M^{\mathsf{B}}_{i^*}, \sigma_{i^*})$ for the player $P^{(i^*)}$ as follows:
   - Let $M^{\mathsf{B}}_{i^*} = M$.
   - Using *public randomness*, Bob picks σ_{i^*} to be a *uniformly random* injection from $[\frac{2n}{k}]$ to $[n + \frac{n}{k}]$.
   - Let V_{i^*} be the image of σ_{i^*} (i.e., $V_{i^*} = \{\sigma_{i^*}(j) \mid j \in [\frac{2n}{k}]\}$).
2. Alice generates the inputs for all other players. Using *private randomness*, Alice first randomly partitions the set $[n + \frac{n}{k}] \setminus V_{i^*}$ into $(k-1)$ sets $\{V'_i\}_{i \in [k] \setminus \{i^*\}}$, where each V'_i has size $\frac{n}{k}$. She then generates the input of each player $P^{(i)}$ $(i \neq i^*)$ as follows:
   - If $f_{i^*}(i) = \mathsf{Yes}$ (resp. $f_{i^*}(i) = \mathsf{No}$), Alice draws a $\mathsf{BHM}^0_{\frac{2n}{k}}$ instance $(M^{\mathsf{B}}_i, x^{\mathsf{B}}_i)$ from $\mathcal{D}^{\mathsf{Y}}_{\mathsf{BHM}}$ (resp. from $\mathcal{D}^{\mathsf{N}}_{\mathsf{BHM}}$).
   - The mapping $\sigma_i : [\frac{2n}{k}] \to [n + \frac{n}{k}]$ is defined as follows. For the $\frac{n}{k}$ entries in $[\frac{2n}{k}]$ where x_i is 0, Alice assigns a *uniformly random* bijection to V'_i. For each entry j in $[\frac{2n}{k}]$ where $x^{\mathsf{B}}_i(j) = 1$, suppose $x^{\mathsf{B}}_i(j)$ is the ℓ-th 1 of x_i, Alice assigns

$\sigma_i(j) = \sigma_{i^*}(j')$ where j' is the index such that $x(j')$ is the ℓ-th 1 of x.[a]

3. Bob runs π_{SMS} for the i^*-th player and Alice runs π_{SMS} for all other players and sends the messages of all other players to Bob.
4. After receiving the messages from Alice, Bob runs the referee part of the protocol π_{SMS}, and outputs the same answer as π_{SMS}.

[a] Recall that x is the input vector to Alice in a BHM^0 instance.

It is straightforward to verify that the distribution of the instances created by this reduction and the original distributions $(\mathcal{D}_{\mathsf{SMS}} \mid f_{i^*})$ and $(\mathcal{D}_{\mathsf{SMS}} \mid g_{i^*})$ are identical. Formally,

Claim 3.6.3. *Suppose (M, x) is a* Yes *(resp.* No*)* BHM *instance; then the* SMS *instance constructed by Alice and Bob is sampled from $\mathcal{D}_{\mathsf{SMS}} \mid f_{i^*}$ (resp. $\mathcal{D}_{\mathsf{SMS}} \mid g_{i^*}$).*

We now use this to prove Theorem 3.8.

Proof of Theorem 3.8. Let $\gamma = \frac{1-2\delta}{2k}$; we first argue that π_{BHM} outputs a correct answer for $\mathsf{BHM}^0_{\frac{2n}{k}}$ w.p. at least $\frac{1}{2}+\gamma$. If the input BHM^0 instance (M, x) is a Yes (resp. No) instance, then by Claim 3.6.3, the distribution of the SMS instance created in π_{BHM} is exactly $\mathcal{D}_{\mathsf{SMS}} \mid f_{i^*}$ (resp. $\mathcal{D}_{\mathsf{SMS}} \mid g_{i^*}$); consequently, π_{SMS} outputs the correct answer w.p. $\frac{1}{2} \cdot \left(p^{\mathsf{Y}}_{f_{i^*}} + p^{\mathsf{N}}_{g_{i^*}}\right)$. Since i^* is a $\frac{1-2\delta}{2k}$-informative instance, we have $\frac{1}{2} \cdot \left(p^{\mathsf{Y}}_{f_{i^*}} + p^{\mathsf{N}}_{g_{i^*}}\right) \geq \frac{1}{2} + \frac{1-2\delta}{2k} = \frac{1}{2} + \gamma$ and hence the protocol Π_{BHM} outputs the correct answer w.p. at least $\frac{1}{2}+\gamma$.

Now notice that in π_{BHM}, Alice is sending messages of $k-1$ players in π_{SMS} to Bob and hence communication cost of π_{BHM} is at most the communication cost of π_{SMS}. Since solving $\mathsf{BHM}_{\frac{2n}{k}}$ on $\mathcal{D}_{\mathsf{BHM}}$ w.p. of success $\frac{1}{2}+\gamma$ requires at least $\Omega(\gamma \cdot \sqrt{\frac{n}{k}})$ communication by Corollary 3.5, we have $\|\pi_{\mathsf{SMS}}\| = \Omega(\gamma \cdot \sqrt{\frac{n}{k}})$. Moreover, $\gamma = \frac{\varepsilon}{k}$ for some constant $\varepsilon > 0$ (since $\delta > 1/2$), hence we obtain that $\|\pi_{\mathsf{SMS}}\| = \Omega\left(\frac{\sqrt{n}}{k\sqrt{k}}\right)$, finalizing the proof. □ Theorem 3.8

3.6.2. An $\Omega(n/\alpha^2)$ Lower Bound for Dense Graphs

We now switch to the dense graphs case (i.e., Part (2) of Result 3.3), and establish a better lower bound of $\Omega(n/\alpha^2)$ for computing an α-estimation to matching in dynamic streams.

We define $\mathsf{EstMatching}_{n,k,\alpha}$ as the *k-player communication* problem of estimating the matching size to within a factor of α, when edges of an n-vertex input graph $G(V, E)$ are partitioned across the k-players. In this section, we prove the following lower bound on the information complexity of $\mathsf{EstMatching}_{n,k,\alpha}$ in the simultaneous communication model.

Theorem 3.9. *For any sufficiently large n and α, there exists some $k = \alpha \cdot \left(\frac{n}{\alpha}\right)^{o(1)}$ such that for any constant $\delta < \frac{1}{2}$, the δ-error simultaneous communication complexity of*

$\mathsf{EstMatching}_{n,k,\alpha}$ *is* $\mathsf{R}^{||}_{\delta}(\mathsf{EstMatching}_{n,k,\alpha}) = k \cdot \Omega(n/\alpha^2)$.

Theorem 3.9, together with the fact that simultaneous communication complexity of a k-player problem lower bounds (up to factor k) the space complexity of single-pass streaming algorithms in dynamic streams [230, 14] (Proposition 2.7.11), implies the $\Omega(n/\alpha^2)$ lower bound in Part (2) of Result 3.3. We now prove Theorem 3.9.

Consider the following distribution $\mathcal{D}_{\mathsf{EM}}$ for $\mathsf{EstMatching}_{n,k,\alpha}$.

The hard distribution $\mathcal{D}_{\mathsf{EM}}$ for $\mathsf{EstMatching}_{n,k,\alpha}$:

Parameters: $r = N^{1-o(1)}$, $t = \frac{\binom{N}{2}-o(N^2)}{r}$, $k = \frac{(\alpha+1)N}{r}$, $n = N + 2k \cdot r$.

1. Fix an (r,t)-RS graph G^{RS} on N vertices with induced matchings $M_1^{\mathsf{RS}}, \ldots, M_t^{\mathsf{RS}}$.
2. Pick $j^* \in [t]$ and $\theta \in \{0,1\}$ independently and uniformly at random.
3. For each player $P^{(i)}$ independently,
   (a) Denote by G_i the input graph of $P^{(i)}$, initialized to be a copy of G^{RS} with vertices $V_i = [N]$. Moreover, define V_i^* as the set of vertices incident on the matching $M_{j^*}^{\mathsf{RS}}$.
   (b) Let $x^{(i)}$ be a t-dimensional vector, whereby $x^{(i)}(j^*) = \theta$ and for any $j \neq j^*$, $x^{(i)}(j)$ is chosen uniformly at random from $\{0,1\}$.
   (c) For any $j \in [t]$, if $x^{(i)}(j) = 0$ *remove* the matching $M_{j^*}^{\mathsf{RS}}$ from G_i (otherwise, do nothing).
4. Pick a random permutation σ of $[n]$. For every player $P^{(i)}$, for each vertex v in $V_i \setminus V_i^*$ with label j ($\in [N]$), *relabel* v to $\sigma(j)$. Enumerate the vertices in V_i^* (from the one with the smallest label to the largest), and relabel the j-th vertex to $\sigma(N+(i-1)\cdot 2r+j)$. In the final graph, the vertices with the same label correspond to the same vertex.

The vertices whose labels belong to $\sigma([N])$ are referred to as *shared* vertices since they belong to the input graph of *every* player, and the vertices V_i^* are referred to as the *private* vertices of the player $P^{(i)}$ since they only appear in the input graph of $P^{(i)}$ (in the final graph, i.e., after relabeling). We point out that, in general, the final graph constructed by this distribution is a multi-graph with n vertices and $O(kN^2) = O(n^2/\alpha)$ edges (counting the multiplicities); the multiplicity of each edge is also at most k. Finally, the existence of an (r,t)-RS graph G^{RS} with the parameters used in this distribution is guaranteed by a result of [18] (see Proposition 2.3.1).

Fix any deterministic δ-error protocol π_{EM} for $\mathsf{EstMatching}_{n,k,\alpha}$ on $\mathcal{D}_{\mathsf{EM}}$. We are going

to prove that $\|\pi_{\mathsf{EM}}\| = \Omega(n/\alpha^2)$. This implies that $\mathsf{D}^{\|}_{\mathcal{D}_{\mathsf{EM}},\delta}(\mathsf{EstMatching}_{n,k,\alpha}) = \Omega(n/\alpha^2)$ which in turns by the easy direction of Yao's minimax principle (see Proposition 2.7.2) would imply Theorem 3.9.

Claim 3.6.4. *Let:*

$$\mathrm{opt}_1 := \min_G \Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{EM}} \text{ conditioned } \textit{on } \theta = 1\Big)$$
$$\mathrm{opt}_0 := \max_G \Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{EM}} \text{ conditioned } \textit{on } \theta = 0\Big).$$

then, $\mathrm{opt}_1 > \alpha \cdot \mathrm{opt}_0$.

Proof. Notice that in each graph G_i, except for the matching $M^{\mathsf{RS}}_{j^*}$, all other matching edges are incident on the set of shared vertices. This implies that across the players, the total contribution of all matchings except for $M^{\mathsf{RS}}_{j^*}$'s is at most N. Consequently, when $\theta = 0$, i.e., when the matching $M^{\mathsf{RS}}_{j^*}$ of each player is removed, $\mathsf{MM}(G) \leq N$. On the other hand, when $\theta = 1$, since the matching $M^{\mathsf{RS}}_{j^*}$ of each player is incident on a unique set of vertices of G (i.e., private vertices), they form a matching of size $k \cdot r = (\alpha+1) \cdot N$. Hence, $\mathsf{MM}(G) \geq (\alpha+1) \cdot N$ in this case. □

Claim 3.6.4 shows that any δ-error protocol π_{EM} for $\mathsf{EstMatching}_{n,k,\alpha}$ can determine the value of the parameter θ in the distribution $\mathcal{D}_{\mathsf{EM}}$ (also with error prob. δ). We use this fact to prove a lower bound on the mutual information between θ and the message of the players. Define Θ, Σ, and J, as random variables for, respectively, θ, the random permutation σ, and the index j^* in the distribution. We have the following simple claim.

Claim 3.6.5. $\mathbb{I}(\Theta\,; \Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) = \Omega(1)$.

Proof. As proven in Claim 3.6.4, protocol π_{EM} can be used directly to determine the value of Θ w.p. $1-\delta$. Hence, by Fano's inequality (Fact 2.6.2), $\mathbb{H}(\Theta \mid \Pi_{\mathsf{EM}}) \leq H_2(\delta)$, since π_{EM} uses only the message Π_{EM} to output the answer. We further have,

$$\begin{aligned} H_2(\delta) &\geq \mathbb{H}(\Theta \mid \Pi_{\mathsf{EM}}) \geq \mathbb{H}(\Theta \mid \Pi_{\mathsf{EM}}, \Sigma, \mathsf{J}) \\ &\qquad \text{(conditioning can only reduce the entropy (Fact 2.6.1-(3)))} \\ &= \mathbb{H}(\Theta \mid \Sigma, \mathsf{J}) - \mathbb{I}(\Theta\,; \Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) = 1 - \mathbb{I}(\Theta\,; \Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}), \end{aligned}$$

where the last equality is because Θ is chosen uniformly at random from $\{0,1\}$ independent of Σ and J. Consequently, we have $\mathbb{I}(\Theta\,; \Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) \geq 1 - H_2(\delta) = \Omega(1)$ (since $\delta < 1/2$). □

We now use the bound in Claim 3.6.5 to lower bound the communication cost of π_{EM}.

Lemma 3.6.6. *Communication cost of π_{EM} is $\|\pi_{\mathsf{EM}}\| = \Omega(t)$.*

Proof. By Claim 3.6.5, $\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) = \Omega(1)$; in the following, we prove that for this to happen, π_{EM} needs to communicate $\Omega(t)$ bits. We have,

$$\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) = \mathop{\mathbb{E}}_{j\in[t]}[\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}=j)] = \frac{1}{t}\sum_{j=1}^{t}\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}=j)$$
$$= \frac{1}{t}\sum_{j=1}^{t}\mathbb{I}(\mathsf{Y}_j\,;\Pi^{(1)}_{\mathsf{EM}},\ldots,\Pi^{(k)}_{\mathsf{EM}} \mid \Sigma, \mathsf{J}=j).$$

Here, for any $j \in [t]$, $\mathsf{Y}_j := (\mathsf{X}_{1,j}, \mathsf{X}_{2,j}, \ldots, \mathsf{X}_{k,j})$, where $\mathsf{X}_{i,j}$ (for any $i \in [k]$) is the random variable denoting $x^{(i)}(j)$. This equality holds since conditioned on $\mathsf{J} = j$, for any $i \in [k]$, each $x^{(i)}(j)$ is assigned to be θ (i.e., $\Theta = \mathsf{X}_{i,j}$ conditioned on $\mathsf{J} = j$) . Moreover,

$$\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) = \frac{1}{t}\sum_{j=1}^{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{Y}_j\,;\Pi^{(i)}_{\mathsf{EM}} \mid \Pi^{<i}_{\mathsf{EM}}, \Sigma, \mathsf{J}=j) \le \frac{1}{t}\sum_{j=1}^{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{Y}_j\,;\Pi^{(i)}_{\mathsf{EM}} \mid \Sigma, \mathsf{J}=j),$$

by Proposition 2.6.4 since for any $i \in [k]$, $\Pi^{(i)}_{\mathsf{EM}} \perp \Pi^{-i}_{\mathsf{EM}} \mid \mathsf{Y}_j, \Sigma, \mathsf{J} = j$ and hence conditioning on $\Pi^{<i}_{\mathsf{EM}}$ can only decrease the mutual information in the above term.

For any player $P^{(i)}$, define Σ_i as the random variable that denotes how the vertices of the graph G_i chosen for player $P^{(i)}$ map to G according to Σ, i.e., specify the labels of vertices. We claim that, $\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) \le \frac{1}{t}\sum_{j=1}^{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{Y}_j\,;\Pi^{(i)}_{\mathsf{EM}} \mid \Sigma_i, \mathsf{J}=j)$. To see this, define Σ_{-i} as the "remainder" of Σ such that $\Sigma = (\Sigma_i, \Sigma_{-i})$. We have $\Pi^{(i)}_{\mathsf{EM}} \perp \Sigma_{-i} \mid \Sigma_i, \mathsf{J} = j$ as $\Pi^{(i)}_{\mathsf{EM}}$ is only a function of the input given to player $P^{(i)}$ and is hence independent of the labeling of vertices in other players input after conditioning on $\sigma_i, \mathsf{J} = j$. As such, by Proposition 2.6.4, removing the conditioning on Σ_{-i} in the previous equation can only increase the mutual information.

We can also drop the conditioning on the event $\mathsf{J} = j$ and have, $\mathbb{I}(\Theta\,;\Pi_{\mathsf{EM}} \mid \Sigma, \mathsf{J}) \le \frac{1}{t}\sum_{j=1}^{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{Y}_j\,;\Pi^{(i)}_{\mathsf{EM}} \mid \Sigma_i)$. The reason is as follows. Firstly, $\Pi^{(i)}_{\mathsf{EM}}$ is a function of (X_i, Σ_i) where $\mathsf{X}_i := (\mathsf{X}_{i,1}, \ldots, \mathsf{X}_{i,t})$ is a random variable for the vector $x^{(i)}$. Moreover, X_i defines the graph G_i without the labels, i.e., over the set of vertices $V_i := [N]$ and Σ_i is the random variable denoting how the vertices of the player $P^{(i)}$ map to G, i.e., specify the labels of vertices. Therefore (X_i, Σ_i) is independent of $\mathsf{J} = j$ (given the input graph G_i, each matching has the same probability of being the chosen matching for j^*); hence it is easy to see that all three random variables in above term are independent of the event $\mathsf{J} = j$. Hence, dropping the event $\mathsf{J} = j$ does not change the term above.

Moreover, since Y_j and Y^{-j} are independent of each other, even conditioned on Σ_i, by

conditioning on $\mathsf{Y}^{<j}$ below we can only increase the mutual information (Proposition 2.6.3),

$$\begin{aligned}
\mathbb{I}(\Theta\ ;\Pi_{\mathsf{EM}}\mid \Sigma,\mathsf{J}) &\leq \frac{1}{t}\sum_{i=1}^{k}\sum_{j=1}^{t}\mathbb{I}(\mathsf{Y}_j\ ;\Pi^{(i)}_{\mathsf{EM}}\mid \mathsf{Y}^{<j},\Sigma_i)\\
&\underset{\text{Fact 2.6.1-(6)}}{=} \frac{1}{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{Y}_1,\ldots,\mathsf{Y}_t\ ;\Pi^{(i)}_{\mathsf{EM}}\mid \Sigma_i)\\
&= \frac{1}{t}\sum_{i=1}^{k}\mathbb{I}(\mathsf{X}_1,\ldots,\mathsf{X}_k\ ;\Pi^{(i)}_{\mathsf{EM}}\mid \Sigma_i)\\
&\quad (\mathsf{Y}_1,\ldots,\mathsf{Y}_t \text{ uniquely determines } \mathsf{X}_1,\ldots,\mathsf{X}_k \text{ and vice versa})\\
&\underset{\text{Fact 2.6.1-(1)}}{\leq} \frac{1}{t}\sum_{i=1}^{k}\left|\Pi^{(i)}_{\mathsf{EM}}\right| = \frac{1}{t}\cdot\|\pi_{\mathsf{EM}}\|.
\end{aligned}$$

As by Claim 3.6.5, $\mathbb{I}(\Theta\ ;\Pi_{\mathsf{EM}}\mid \Sigma,\mathsf{J}) = \Omega(1)$, we have that $\|\pi_{\mathsf{EM}}\| = \Omega(t)$. □ Lemma 3.6.6

Theorem 3.9 now follows from Lemma 3.6.6 by noticing that $n = (2\alpha+1)\cdot N$, $r = 2\alpha N/k$ and $t \geq \frac{N^2}{2r} = \frac{N\cdot k}{2\alpha} = \Omega(nk/\alpha^2)$.

3.7. Space Lower Bounds for $(1+\varepsilon)$-Estimating Matching Size

In this section, we present our space lower bounds for algorithms that compute a $(1+\varepsilon)$-approximation of the maximum matching size in graph streams. We first introduce some notation which will be used throughout this section.

Notation. Fix any (r,t)-RS graph $G^{\mathsf{RS}}(V,E)$ (for any parameters r,t) with induced matchings $M^{\mathsf{RS}}_1,\ldots,M^{\mathsf{RS}}_t$. For each matching M^{RS}_i, we assume an arbitrary ordering of the edges in M^{RS}_i, denoted by $e_{i,1},\ldots,e_{i,r}$, and further denote $e_{i,j} := (u_{i,j},v_{i,j})$ for all $j\in[r]$. Let $L(M^{\mathsf{RS}}_i) := \{u_{i,1},\ldots,u_{i,r}\}$ and $R(M^{\mathsf{RS}}_i) := \{v_{i,1},\ldots,v_{i,r}\}$. We emphasize that we do not require $G^{\mathsf{RS}}(V,E)$ to be necessarily a *bipartite* graph; each bipartition $L(M^{\mathsf{RS}}_i)$ and $R(M^{\mathsf{RS}}_i)$ (for $i\in[t]$) is defined locally for the matching itself and hence a vertex v is allowed to belong to, say, $L(M^{\mathsf{RS}}_i)$ and $R(M^{\mathsf{RS}}_j)$ for $i\neq j$, simultaneously.

Furthermore, for each matching M^{RS}_i and any boolean vector $x\in\{0,1\}^r$, we define the matching $M^{\mathsf{RS}}_i|_x$ as the subset of (the edges) of M^{RS}_i obtained by retaining the edge $e_{i,j}\in M^{\mathsf{RS}}_i$ (for any $j\in[r]$) iff $x(j)=1$. In addition, for the vertex set $R(M^{\mathsf{RS}}_i)$ and any perfect p-hypermatching[3] $\mathcal{M}$ on $[r]$, we define the *p-clique family* of $\mathcal{M}$ on $R(M^{\mathsf{RS}}_i)$ to be a set of $|\mathcal{M}|$ cliques where the vertices $C_{\mathbf{e}}$ of each clique is defined by a distinct hyperedge $\mathbf{e}\in\mathcal{M}$: $C_{\mathbf{e}} := \{v_{i,k}\mid k\in\mathbf{e}\}$.

[3]Throughout this section, we use p instead of the usual parameter t for hypermatchings in order to avoid confusion with the parameter t in RS graphs

3.7.1. Insertion-Only Streams

We define $\mathsf{EstMatching}_{n,\varepsilon}$ as the *two-player one-way communication* problem of estimating the matching size to within a factor of $(1+\varepsilon)$, when Alice and Bob are each given a subset of the edges of an n-vertex input graph $G(V,E)$. In this section, we prove the following lower bound on the communication complexity of $\mathsf{EstMatching}_{n,\varepsilon}$.

Theorem 3.10. *For large n and small $\varepsilon < 1/2$, the δ-error one-way communication complexity of $\mathsf{EstMatching}_{n,\varepsilon}$ for constant $\delta < 1/2$ is $\mathsf{R}^{\text{1-way}}_{\delta}(\mathsf{EstMatching}_{n,\varepsilon}) = \mathsf{RS}(n) \cdot n^{1-O(\varepsilon)}$.*

Similar to the previous section, this result together with the fact that one-way communication complexity lower bounds space complexity of single-pass insertion-only streaming algorithms (Proposition 2.7.10), implies the $\mathsf{RS}(n) \cdot n^{1-O(\varepsilon)}$ space lower bound in Part-(1) of Result 3.4. We now prove Theorem 3.10.

Suppose the maximum value for $\mathsf{RS}(n)$ is achieved by an (r,t)-RS graph with the parameter $r = \mathsf{c}_{\text{rs}} \cdot n$. We propose the following (hard) input distribution $\mathcal{D}_{\mathsf{M}}$ for $\mathsf{EstMatching}_{n,\varepsilon}$.

The hard distribution $\mathcal{D}_{\mathsf{M}}$ for $\mathsf{EstMatching}_{n,\varepsilon}$:

Parameters: $N := \frac{n}{2-2\mathsf{c}_{\text{rs}}}$, $r := \mathsf{c}_{\text{rs}} \cdot N$, $t := \mathsf{RS}(N)$, and $p := \left\lfloor \frac{\mathsf{c}_{\text{rs}}}{2\varepsilon} \right\rfloor$.

- The input to the players is a graph $G(V, E_A \cup E_B)$ where E_A is given to Alice and E_B is given to Bob.
- **Alice:**
  1. Let V_1 ($\subset V$) and $V_2 := V \setminus V_1$ be, respectively, a set of N and $n-N$ vertices.
  2. Let H be any fixed (r,t)-RS graph with $V(H) = V_1$.
  3. Draw r-dimensional binary vectors $x^{(1)}, \ldots, x^{(t)}$ *independently* following the distribution $\mathcal{D}_{\mathsf{BHH}}$ for $\mathsf{BHH}^0_{r,p}$.
  4. The input to Alice is the edge-set $E_A := M_1 \cup \ldots \cup M_t$, where $M_j := M_j^{\mathsf{RS}}|_{x^{(j)}}$.
- **Bob:**
  1. Pick $j^* \in [t]$ uniformly at random.
  2. For the vector $x^{(j^*)}$, draw a perfect p-hypermatching $\mathcal{M}$ following the distribution $\mathcal{D}_{\mathsf{BHH}}$ conditioned on $x^{(j^*)}$; consequently, $(x^{(j^*)}, \mathcal{M})$ is a $\mathsf{BHH}^0_{r,p}$ instance drawn from the distribution $\mathcal{D}_{\mathsf{BHH}}$.
  3. Let $E_{B,1}$ be an arbitrary perfect matching between $V_1 \setminus V(M_{j^*}^{\mathsf{RS}})$ and V_2.
  4. Let $E_{B,2}$ be the edges of the p-clique family of $\mathcal{M}$ on $R(M_{j^*}^{\mathsf{RS}})$.
  5. The input to Bob is the edge-set $E_B := E_{B,1} \cup E_{B,2}$.

We say that the instance $(x^{(j^*)}, \mathcal{M})$ of $\mathsf{BHH}^0_{r,p}$ in the distribution (denoted by I_{BHH}) is *embedded* inside $\mathcal{D}_{\text{MM}}$. The following claim established the connection between I_{BHH} and

maximum matching size in G.

Claim 3.7.1. *Let:*

$$\mathrm{opt}_{\mathsf{Yes}} := \min_{G}\Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{M}} \text{ conditioned } \textit{on } I_{\mathsf{BHH}} \textit{ being a } \mathsf{Yes} \textit{ instance}\Big)$$
$$\mathrm{opt}_{\mathsf{No}} := \max_{G}\Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{M}} \text{ conditioned } \textit{on } I_{\mathsf{BHH}} \textit{ being a } \mathsf{No} \textit{ instance}\Big)$$

then, $(1-\varepsilon)\cdot \mathrm{opt}_{\mathsf{Yes}} > \mathrm{opt}_{\mathsf{No}}$.

Proof. Let M^* be a maximum matching in G. Since all vertices in V_2 have degree 1, without loss of generality, we can assume M^* contains the matching $E_{B,1}$ between $V_1 \setminus V(M^{\mathsf{RS}}_{j^*})$ and V_2. Consequently the size of M^* only depends on how many vertices in $V(M^{\mathsf{RS}}_{j^*})$ can be matched with each other.

Consider the subgraph $H := G[L(M^{\mathsf{RS}}_{j^*}) \cup R(M^{\mathsf{RS}}_{j^*})]$ of G; by Claim 3.3.4, if I_{BHH} is a Yes instance, then $\mathsf{MM}(H) = \frac{3r}{4}$. Hence, in this case,

$$\mathsf{MM}(G) = |V_2| + \mathsf{MM}(H) = N - 2r + \frac{3r}{4} = N - \frac{5\mathsf{c}_{\mathrm{rs}}N}{4} = \frac{4-5\mathsf{c}_{\mathrm{rs}}}{4}\cdot N$$

If I_{BHH} is a No instance, then $\mathrm{opt}(H) = \frac{3r}{4} - \frac{r}{2p}$. Hence, in this case,

$$\mathsf{MM}(G) = |V_2| + \mathsf{MM}(H) \leq N - 2r + \frac{3r}{4} - \frac{r}{2p} = N - \frac{5\mathsf{c}_{\mathrm{rs}}N}{4} - \frac{\mathsf{c}_{\mathrm{rs}}}{2p}N = \frac{4-5\mathsf{c}_{\mathrm{rs}}}{4}\cdot N - \frac{\mathsf{c}_{\mathrm{rs}}}{2p}N$$

The bound on $\mathrm{opt}_{\mathsf{Yes}}$ and $\mathrm{opt}_{\mathsf{No}}$ now follows from the fact that $p \leq \frac{\mathsf{c}_{\mathrm{rs}}}{2\varepsilon}$ and therefore $\frac{\mathsf{c}_{\mathrm{rs}}}{2p}N \geq \varepsilon N > \varepsilon\cdot\big(\frac{4-5\mathsf{c}_{\mathrm{rs}}}{4}\cdot N\big)$. □

Fix any δ-error protocol Π_{Matching} for $\mathsf{EstMatching}_{n,\varepsilon}$ on $\mathcal{D}_{\mathsf{M}}$; Claim 3.7.1 implies that Π_{Matching} is also a δ-error protocol for solving the embedded instance I_{BHH}: simply return Yes whenever the estimate is larger than $\mathrm{opt}_{\mathsf{No}}$ and return No otherwise. We now use this fact to design a protocol Π_{BHH} for solving $\mathsf{BHH}^0_{r,p}$ on $\mathcal{D}_{\mathsf{BHH}}$, and prove that the communication cost of Π_{Matching} is t times the communication complexity of $\mathsf{BHH}^0_{r,p}$.

The protocol Π_{BHH} for reducing $\mathsf{BHH}^0_{r,p}$ to $\mathsf{EstMatching}_{n,\varepsilon}$.

1. Let $(x, \mathcal{M})$ be the input $\mathsf{BHH}^0_{r,p}$ instance (x is given to Alice and $\mathcal{M}$ is given to Bob).
2. Using *public randomness*, Alice and Bob sample an index $j^* \in [t]$ uniformly at random.
3. Let $x^{(1)}, \ldots, x^{(t)}$ be t vectors in $\{0,1\}^r$ whereby $x^{(j^*)} = x$ and for any $j \neq j^*$, $x^{(j)}$ is

sampled by Alice using *private randomness* as in the distribution $\mathcal{D}_{\mathsf{M}}$. Alice creates the edges E_A following the distribution $\mathcal{D}_{\mathsf{M}}$ using these vectors.

4. Given the p-hypermatching $\mathcal{M}$ as input, Bob creates $E_{B,1}$ as an arbitrary perfect matching between $V_1 \setminus V(M^{\mathsf{RS}}_{j^*})$ and V_2. He also creates $E_{B,2}$ as the edges of the p-clique family of $\mathcal{M}$ on $R(M^{\mathsf{RS}}_{j^*})$ (V_1, V_2, and $M^{\mathsf{RS}}_{j^*}$ are defined exactly as in $\mathcal{D}_{\mathsf{M}}$).
5. The players then run Π_{Matching} on the graph $G(V, E_A \cup E_B)$ and Bob outputs Yes if the output is larger than opt_{No} and No otherwise.

The correctness of the protocol follows immediately from Claim 3.7.1. We now bound the information cost of this new protocol and use it to bound its communication cost by Proposition 2.8.3.

Lemma 3.7.2. $\mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{BHH}}}(\Pi_{\mathsf{BHH}}) \leq \frac{1}{t} \cdot \mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{M}}}(\Pi_{\mathsf{Matching}})$.

Proof. Let R denote the public randomness used by Π_{BHH}. We have,

$$\begin{aligned}
\mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{BHH}}}(\Pi_{\mathsf{BHH}}) &= \mathbb{I}(\mathsf{X}\,;\Pi_{\mathsf{BHH}} \mid \mathsf{R}) \quad \text{(by Proposition 2.8.4 as } \Pi_{\mathsf{BHH}} \text{ is a one-way protocol)} \\
&= \mathbb{I}(\mathsf{X}\,;\Pi_{\mathsf{Matching}} \mid \mathsf{J}) \\
&\quad (\mathsf{R} = \mathsf{J} \text{ and the message of } \Pi_{\mathsf{BHH}} \text{ is the same as } \Pi_{\mathsf{Matching}} \text{ after fixing the index } j^*) \\
&= \mathop{\mathbb{E}}_{j \in [t]}[\mathbb{I}(\mathsf{X}\,;\Pi_{\mathsf{Matching}} \mid \mathsf{J} = j)] = \frac{1}{t} \cdot \sum_{j=1}^{t} \mathbb{I}(\mathsf{X}_j\,;\Pi_{\mathsf{Matching}} \mid \mathsf{J} = j) \\
&\quad (\text{distribution of } (\Pi_{\mathsf{Matching}}, \mathsf{X}_j), \text{ conditioned on } \mathsf{J} = j, \text{ is the same under } \mathcal{D}_{\mathsf{M}} \text{ and } \mathcal{D}_{\mathsf{BHH}}) \\
&= \frac{1}{t} \cdot \sum_{j=1}^{t} \mathbb{I}(\mathsf{X}_j\,;\Pi_{\mathsf{Matching}}),
\end{aligned}$$

where the last equality is true since the random variables X_j and Π_{Matching} are both independent of the event $\mathsf{J} = i$ (by definition of the distribution $\mathcal{D}_{\mathsf{M}}$) and hence we can drop the conditioning. Finally,

$$\begin{aligned}
\mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{BHH}}}(\Pi_{\mathsf{BHH}}) = \frac{1}{t} \cdot \sum_{j=1}^{t} \mathbb{I}(\mathsf{X}_j\,;\Pi_{\mathsf{Matching}}) &\leq \frac{1}{t} \cdot \sum_{j=1}^{t} \mathbb{I}(\mathsf{X}_j\,;\Pi_{\mathsf{Matching}} \mid \mathsf{X}^{<j}) \\
&\quad (\text{since } \mathsf{X}_j \perp \mathsf{X}^{<j} \text{ and by Proposition 2.6.3}) \\
&\underset{\text{Fact 2.6.1-(6)}}{=} \mathbb{I}(\mathsf{X}_1, \ldots, \mathsf{X}_t\,;\Pi_{\mathsf{Matching}}) = \frac{1}{t} \cdot \mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{M}}}(\Pi_{\mathsf{Matching}}).
\end{aligned}$$

where the second last inequality is again by Proposition 2.8.4 as $\mathsf{X}_1, \ldots, \mathsf{X}_t$ uniquely determines the input E_A to Alice in Π_{Matching}. □

Theorem 3.10 now follows from Lemma 3.7.2, lower bound of $\Omega(r^{1-1/p}) = n^{1-O(\varepsilon)}$ for information complexity of $\mathsf{BHH}^0_{r,p}$ in Corollary 3.5, and the choice of $t = \mathsf{RS}(n)$ (and applying Proposition 2.8.3 to extend the information cost to a communication lower bound).

3.7.2. Dynamic Streams

We define $\mathsf{EstMatching}_{n,k,\varepsilon}$ as the *k-player simultaneous communication* problem of estimating the maximum matching size to within a factor of $(1+\varepsilon)$, when edges of an n-vertex input graph $G(V,E)$ are partitioned across the k-players and the referee (see Remark 2.7.8). In this section, we prove the following lower bound on the communication complexity of $\mathsf{EstMatching}_{n,k,\varepsilon}$ in the simultaneous communication model.

Theorem 3.11. *For any sufficiently large n and sufficiently small $\varepsilon < \frac{1}{2}$, there exists some $k = n^{o(1)}$ such that the δ-error k-party simultaneous communication complexity of $\mathsf{EstMatching}_{n,k,\varepsilon}$ for constant $\delta < 1/2$ is $\mathsf{R}^{||}_{\delta}(\mathsf{EstMatching}_{n,k,\varepsilon}) = n^{2-O(\varepsilon)}$.*

Similar to before, Theorem 3.11 together with the fact that simultaneous communication complexity lower bounds space complexity of single-pass dynamic streaming algorithms (Proposition 2.7.11), implies the $n^{2-O(\varepsilon)}$ space lower bound in Part-(2) of Result 3.4 (as $k = n^{o(1)}$ in Theorem 3.11). We now prove Theorem 3.11.

We propose the following (hard) distribution $\mathcal{D}_{\mathsf{M}}$ for $\mathsf{EstMatching}_{n,k,\varepsilon}$.

The hard distribution $\mathcal{D}_{\mathsf{M}}$ for $\mathsf{EstMatching}_{n,k,\varepsilon}$:

Parameters: $r = N^{1-o(1)}$, $t = \frac{\binom{N}{2}-o(N^2)}{r}$, $k = \frac{N}{\varepsilon \cdot r}$, $n = N + k \cdot r$, and $p := \lfloor \frac{1}{8\varepsilon} \rfloor$.

1. Fix an (r,t)-RS graph G^{RS} on N vertices.
2. Pick $j^* \in [t]$ uniformly at random and draw a $\mathsf{BHH}^0_{r,p}$ instance $(x^{(j^*)}, \mathcal{M})$ from the distribution $\mathcal{D}_{\mathsf{BHH}}$.
3. For each player $P^{(i)}$ independently,
   (a) Denote by G_i the input graph of $P^{(i)}$, initialized to be a copy of G^{RS} with vertices $V_i = [N]$.
   (b) Let V_i^* be the set of vertices matched in the j^*-th induced matching of G_i. Change the induced matching $M^{\mathsf{RS}}_{j^*}$ of G_i to $M_{j^*} := M^{\mathsf{RS}}_{j^*}|_{x^{(j^*)}}$.
   (c) For any $j \in [t] \setminus \{j^*\}$, draw a vector $x^{(i,j)} \in \{0,1\}^r$ following the distribution $\mathcal{D}_{\mathsf{BHH}}$ for $\mathsf{BHH}^0_{r,p}$, and change the induced matching M^{RS}_j of G_i to $M_j := M^{\mathsf{RS}}_j|_{x^{(j)}}$.
   (d) Create the p-clique family of $\mathcal{M}$ on the vertices $R(M^{\mathsf{RS}}_{j^*})$, and give the edges of the p-clique family to the referee.
4. Pick a random permutation σ of $[n]$. For every player $P^{(i)}$, for each vertex v in

$V_i \setminus V_i^*$ with label j ($\in [N]$), *relabel* v to $\sigma(j)$. Enumerate the vertices in V_i^* (from the one with the smallest label to the largest), and relabel the j-th vertex to $\sigma(N+(i-1)\cdot 2r+j)$. In the final graph, the vertices with the same label correspond to the same vertex.

The vertices whose labels belong to $\sigma([N])$ are referred to as *shared* vertices since they belong to the input graph of *every* player, and the vertices V_i^* are referred to as the *private* vertices of the player $P^{(i)}$ since they only appear in the input graph of $P^{(i)}$ (in the final graph, i.e., after relabeling). We point out that, in general, the final graph constructed by this distribution is a multi-graph with n vertices and $O(kN^2) = O(n^2)$ edges (counting the multiplicities); the multiplicity of each edge is also at most k. Finally, the existence of an (r,t)-RS graph G^{RS} with the parameters used in this distribution is guaranteed by a result of [18] (see Proposition 2.3.1).

Similar to the lower bound in Section 3.7.1, let I_{BHH} be the *embedded* $\mathsf{BHH}^0_{r,p}$ instance $(x^{(i)}, \mathcal{M})$. The following claim is analogous to Claim 3.7.1 in Section 3.7.1.

Claim 3.7.3. *Let:*

$$\mathrm{opt}_{\mathsf{Yes}} := \min_G \Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{M}} \text{ conditioned } \textit{on } I_{\mathsf{BHH}} \textit{ being a } \mathsf{Yes} \textit{ instance}\Big)$$
$$\mathrm{opt}_{\mathsf{No}} := \max_G \Big(\mathsf{MM}(G) \mid G \textit{ is chosen from } \mathcal{D}_{\mathsf{M}} \text{ conditioned } \textit{on } I_{\mathsf{BHH}} \textit{ being a } \mathsf{No} \textit{ instance}\Big)$$

then, $(1-\varepsilon)\cdot \mathrm{opt}_{\mathsf{Yes}} > \mathrm{opt}_{\mathsf{No}}$.

Proof. We partition the edges of G into $k+1$ groups: for any $i \in [k]$, group i contains the edges that are between the private vertices V_i^* of player $P^{(i)}$, and group $k+1$ contains the edges incident on at least one shared vertex. Let $H_i := G[V_i^*]$, i.e., the subgraph of G induced on the vertices V_i^*.

If I_{BHH} is a Yes instance, then for any $i \in [k]$, $\mathrm{opt}(H_i) = \frac{3r}{4}$ by Claim 3.3.4. Since V_i^* are private vertices, one can choose *any* matching from each H_i, and the collection of the chosen edges form a matching of G. Therefore, $\mathsf{MM}(G) > \sum_{i=1}^{k} \mathsf{MM}(H_i) = \frac{3kr}{4} = \frac{3N}{\varepsilon}$. Note that, $\mathsf{MM}(G)$ is *strictly* larger than $\frac{3N}{\varepsilon}$ since one can add (any) edge between the public vertices to the matching.

If I_{BHH} is a No instance, then $\mathsf{MM}(H_i) = \frac{3r}{4} - \frac{r}{2p}$. Since $\mathsf{MM}(G)$ is at most the summation of the maximum matching size in each group, we have $\mathsf{MM}(G) \leq \sum_{i=1}^{k} \mathsf{MM}(H_i) + N \leq \frac{3kr}{4} - \frac{kr}{2p} + N \leq \frac{3N}{\varepsilon} - 3N$, and the gap between $\mathrm{opt}_{\mathsf{Yes}}$ and $\mathrm{opt}_{\mathsf{No}}$ follows. □

Fix any δ-error protocol Π_{Matching} for $\mathsf{EstMatching}_{n,k,\varepsilon}$ on $\mathcal{D}_{\mathsf{M}}$; Claim 3.7.3 implies that

Π_{Matching} is also a δ-error protocol for solving the embedded instance I_{BHH}: simply return Yes whenever the estimate is larger than opt_{No} and return No otherwise. In the following, we use this fact to design a protocol Π_{BHH} for solving $\mathsf{BHH}^0_{r,p}$ on $\mathcal{D}_{\mathsf{BHH}}$, and then prove that the information cost of Π_{Matching} is t times the information cost of $\mathsf{BHH}^0_{r,p}$.

In the protocol Π_{BHH}, Alice will simulate all k players of $\mathsf{EstMatching}_{n,k,\varepsilon}$ and Bob will simulate the referee; Alice and Bob will use public coins to draw the special index j^* and the permutation σ. Together with the input from $\mathcal{D}_{\mathsf{BHH}}$, Alice and Bob will be able to create a $\mathsf{EstMatching}_{n,k,\varepsilon}$ instance. The reduction is formally defined as follows (the parameters used in the reduction are exactly the same as that in the definition of $\mathcal{D}_{\mathsf{M}}$).

The protocol Π_{BHH} for reducing $\mathsf{BHH}^0_{r,p}$ to $\mathsf{EstMatching}_{n,k,\varepsilon}$.

1. Let $(x, \mathcal{M})$ be the input $\mathsf{BHH}^0_{r,p}$ instance (x is given to Alice and $\mathcal{M}$ is given to Bob).
2. Using *public randomness*, Alice and Bob sample an index $j^* \in [t]$, and a permutation σ on $[n]$ uniformly at random.
3. For any player $P^{(i)}$, let $x^{(i,1)}, \ldots, x^{(i,t)}$ be t vectors in $\{0,1\}^r$ whereby $x^{(i,j^*)} = x$ (i.e., Alice's input in the $\mathsf{BHH}^0_{r,p}$ problem) and for any $j \neq j^*$, $x^{(i,j)}$ is sampled by Alice using *private randomness* as in the distribution $\mathcal{D}_{\mathsf{M}}$. Alice then uses these vector together with permutation σ to create the input graph G_i for each player $P^{(i)}$ for $i \in [k]$ following how G_i is created in the distribution $\mathcal{D}_{\mathsf{M}}$ for $\mathsf{EstMatching}_{n,k,\varepsilon}$.
4. The vertices $R(M^{\mathsf{RS}}_{j^*})$ of each player will be mapped (by σ) to a different set of vertices in G. Since Bob knows σ and j^*, and the (input) p-hypermatching $\mathcal{M}$, Bob can create the p-clique families of each player (following referee's input in $\mathcal{D}_{\mathsf{M}}$).
5. The players then run Π_{Matching} on the $\mathsf{EstMatching}_{n,k,\varepsilon}$ that they created, and Bob outputs Yes if the matching size estimate is larger than opt_{No} and No otherwise.

It is straightforward to verify that the distribution of the $\mathsf{EstMatching}_{n,k,\varepsilon}$ instance created by the protocol Π_{BHH} is identical to the distribution $\mathcal{D}_{\mathsf{M}}$. The correctness of the protocol now follows immediately from Claim 3.7.1. The following lemma now bounds the information cost of this protocol. The proof is almost identical to the proof of Lemma 3.6.6 in Section 3.6.2 and hence we do not repeat it again here (see [33] for the detailed proof).

Lemma 3.7.4. $\mathsf{IC}^{\text{ext}}_{\mathcal{D}_{\mathsf{BHH}}}(\Pi_{\mathsf{BHH}}) \leq \frac{1}{t} \cdot \|\Pi_{\mathsf{Matching}}\|$.

Theorem 3.11 now follows from Lemma 3.7.4, the lower bound of $\Omega(r^{1-1/p}) = n^{1-O(\varepsilon)}$ for information cost of $\mathsf{BHH}^0_{r,p}$ in Corollary 3.5, and the choice of $t = \Theta(n)$.

3.8. Further Implications of Our Impossibility results

We conclude this chapter by listing some simple applications of our impossibility results to other problems as well as models of computation and general techniques such as linear sketches and composable coresets studied in this thesis. The results in this part have not been published previously (albeit hinted at in our subsequent paper [30]).

Extension to the minimum vertex cover problem. Our lower bound in Result 3.2 can also be extended to the minimum vertex cover problem implying the following.

Theorem 3.12. *For any $\alpha \leq \sqrt{n}$, any randomized single-pass streaming algorithm that outputs an α-approximate vertex cover in dynamic graph streams with a constant probability requires $\Omega(n^2/\alpha^3)$ space in the worst case.*

Theorem 3.12 follows from the following more general result.

Theorem 3.13. *For any $\alpha \leq \sqrt{n}$, there exists some $k = \alpha \cdot \left(\frac{n}{\alpha}\right)^{o(1)}$ such that the k-party simultaneous communication complexity of finding an α-approximate vertex cover, denoted by $\mathsf{ApxVC}_{n,k,\alpha}$, is $\mathsf{R}^{||}_{1/10}(\mathsf{ApxVC}_{n,k,\alpha}) = k \cdot \Omega(n^2/\alpha^3)$.*

In the following, we briefly give the high level idea behind the proof of Theorem 3.13. Recall the proof of Theorem 3.7 in the simultaneous communication model and the distribution used in the proof of this theorem. Suppose we instead use the following distribution (the difference is only that we are now sampling edges in each induced matching at a rate of $1/\alpha$ instead of $1/2$).

A hard input distribution $\mathcal{D}$ for minimum vertex cover. (parameterized by a sufficiently large integer $N > 0$)

Parameters: $r = N^{1-o(1)}$, $t = \frac{\binom{N}{2} - o(N^2)}{r}$, $k = \frac{10\alpha \cdot N}{r}$, $n = N + 2 \cdot k \cdot r$.

1. Fix an (r,t)-RS graph G^{RS} on N vertices with induced matchings $M_1^{\mathsf{RS}}, \ldots, M_t^{\mathsf{RS}}$.
2. Pick $j^* \in [t]$ uniformly at random.
3. For each player $P^{(i)}$ independently,
   (a) Denote by G_i the input graph of $P^{(i)}$, initialized to be a copy of G^{RS} with vertices $V_i = [N]$. Moreover, define V_i^* as the set of vertices incident on $M_{j^*}^{\mathsf{RS}}$.
   (b) Sample each edge in G_i w.p. $1/\alpha$ independently and drop the remaining edges.
4. Pick a random permutation σ of $[n]$. For every player $P^{(i)}$, for each vertex v in $V_i \setminus V_i^*$ with label ℓ, *relabel* v with $\sigma(\ell)$. Enumerate the vertices in V_i^* (from the

one with the smallest label to the largest), and relabel the ℓ-th enumerated vertex with $\sigma(N + (i-1) \cdot 2r + \ell)$. In the final graph, the vertices with the same label correspond to the same vertex.

It is easy to see that in this distribution, the minimum vertex cover is of size $O(n\alpha)$ by picking all shared vertices as well as one of the endpoints of each edge from the special matchings (note that by our sampling strategy, the total size of all special matchings is now $O(n/\alpha)$ instead of $\Omega(n)$ in the distribution of Theorem 3.7). On the other hand, if the players communicate $o(n^2/\alpha^3)$, then the coordinator is not able to find the edges of special matchings and is hence forced to pick almost all vertices in V_i^* for every player $P^{(i)}$ to ensure that a vertex cover of the graph is chosen, hence resulting in a vertex cover of size $\Omega(n)$. We leave the formal proof of Theorem 3.13 as an exercise but emphasize that the proof follows in a straightforward way from the proof of Theorem 3.7.

Extension to other settings. The following results follow from simple applications of the proof of Results 3.2 (in particular Theorem 3.7 and its aforementioned extension to the minimum vertex cover problem in Theorem 3.13), and the connection between different models cited in Section 1.1.4:

- Any linear sketch or composable coreset for approximating the maximum matching or minimum vertex cover problems to within a factor of $n^{o(1)}$ requires $n^{2-o(1)}$ space.
- Any distributed communication protocol with one round of communication for approximating the maximum matching or minimum vertex cover problems to within a factor of $n^{o(1)}$ requires $n^{2-o(1)}$ total communication.
- Any MPC algorithm with one round of computation in which the output is stored on a single machine that is able to approximate the maximum matching or minimum vertex cover problems to within a factor of $n^{o(1)}$ requires $n^{2-o(1)}$ memory per-machine.

Informally speaking, the above implications suggest that achieving even a weak approximation ratio of $n^{o(1)}$ for the maximum matching problem does not have a better solution than essentially either storing the whole graph as a linear sketch or composable coreset, or communicate it entirely in the distributed communication and MPC models. Similar implications as above hold for the problem of estimating the maximum matching size (instead of finding the actual edges), albeit for the much more accurate approximation of $(1 \pm o(1))$ using Results 3.3 and 3.4.

Chapter 4
A Framework for Optimization on Massive Graphs

In this chapter, we introduce a new framework for graph optimization on massive graphs and apply it to two prominent problems of maximum matching and minimum vertex cover. The materials in this chapter are based on [30, 29] with further simplifications.

A basic and abstract algorithmic approach to large-scale optimization is as follows: partition the data into multiple pieces, compute a representation or a summary of each piece, merge the summaries together and recover the solution from the merge without further accessing the original input. As we argued in Section 1.1.4, various algorithmic approaches for processing massive datasets—most notably linear sketches and composable coresets—in streaming, distributed communication, and MPC models fall under this category. Indeed, successful applications of these techniques has yielded numerous efficient algorithms for many graph problems including connectivity, minimum spanning tree, cut and spectral sparsifiers, spanners, densest subgraphs, subgraph counting, and so on (see, e.g., [11, 12, 13, 209, 208, 102, 61, 169, 243, 182] and references therein).

Nevertheless, our results in the previous chapter (see Section 3.8), ruled out the applicability of these techniques for the prominent problems of maximum matching and minimum vertex cover. In this chapter, we introduce a simple tweak to this general strategy and introduce a new framework for graph optimization on massive graphs that can bypass the aforementioned impossibility results (in some cases, by making natural assumptions about the input) and even improve the state-of-the-art in a unified way across multiple models. In the next chapter, we show further modifications of this framework that is tailored to the MPC model and allows for even more efficient algorithms in that model.

HighLights of Our Contributions

In this chapter, we will establish:

- An algorithmic approach for bypassing the impossibility results for matching and vertex cover in streaming, distributed, and MPC models by making natural assumptions about the input partitioning (Section 4.4).
- Lower bounds on the performance of any algorithm using this new approach proving the near optimality of our results in terms of their resource requirement (Section 4.5).
- Better algorithms using this approach that improve the state-of-the-art in a unified way across multiple models in one or all parameters involved (Section 4.6).

4.1. Background

As massive graphs become more prevalent, there is a rapidly growing need for scalable algorithms that solve classical graph problems on large datasets, in particular the streaming, distributed communication model, and the MPC model studied in this thesis. Given the variety of relevant models for processing massive graphs, there has been a lot of attention on designing general algorithmic techniques that can be applicable across a wide range of settings. Two particularly successful techniques in this context are linear sketches and composable coresets introduced in Section 1.1.4. There has been a considerable amount of work in designing linear sketches and composable coresets for optimization problems in recent years [10, 12, 208, 35, 101, 81, 209, 61, 243, 40, 43, 49, 186, 250, 249]. Successful applications of these two techniques has yielded $\widetilde{O}(n)$ size summaries for many graph problems. However, two prominent problems are notably absent from the list of successes, namely, the *maximum matching* problem and the *minimum vertex cover* problem. Indeed, our results from Chapter 3 (in particular Section 3.8) imply that both matching and vertex cover require summaries of size $n^{2-o(1)}$ for even computing a weak $n^{o(1)}$-approximate solution using these techniques, implying that these techniques are essentially useless for these problems.

This state-of-affairs is the starting point for our work in this chapter. Our main insight is that a natural *data oblivious partitioning scheme* completely alters this landscape: both matching and vertex cover problems admit $O(1)$-approximate composable coresets of size $\widetilde{O}(n)$ provided the edges of the graph are ***randomly partitioned*** across multiple pieces (as opposed to linear sketches and composable coresets that assume arbitrarily partitioning of input). The idea that random partitioning of data can help in these scenarios was nicely illustrated in the recent work of [249] on maximizing submodular functions. Our work can be seen as the first illustration of this idea in the domain of graph algorithms. The applicability of this idea to graph problems has been cast as an open problem in [249].

4.1.1. Randomized Composable Coresets

We follow the notation of [249] with a modification to adapt to our application in graphs. Let E be an edge-set of a graph $G(V, E)$; we say that a partition $\left\{E^{(1)}, \ldots, E^{(k)}\right\}$ of the edges E is a *random k-partitioning* iff the sets are constructed by assigning each edge in E independently to a set $E^{(i)}$ chosen uniformly at random. A random partitioning of the edges naturally defines partitioning the graph $G(V, E)$ into k graphs $G^{(1)}, \ldots, G^{(k)}$ whereby $G^{(i)} := G(V, E^{(i)})$ for any $i \in [k]$, and hence we use random partitioning for both the edge-set and the input graph interchangeably.

Definition 4.1 (Randomized Composable Coreset)**.** *For a graph problem P, consider an algorithm* ALG *that takes as input a graph G and returns a* ***subgraph*** $\mathsf{ALG}(G) \subseteq G$*.* ALG *is*

said to output an α-approximate randomized coreset for P if given any graph $G(V,E)$ and a random k-partition of G into $\{G^{(i)}(V,E^{(i)})\}$, $P\left(\mathsf{ALG}(G^{(1)}) \cup \ldots \cup \mathsf{ALG}(G^{(k)})\right)$ is an α-approximation to $P(G)$ with high probability where the probability is taken over the random choice of the k-partitioning (and the algorithm).

*We refer to the **number of edges** in the returned subgraph by ALG as **size** of the coreset.*

We further augment this definition by allowing the coresets to also contain a *fixed solution* to be *directly* added to the final solution of the composed coresets. In this case, size of the coreset is measured both in the number of edges in the output subgraph plus the number of vertices and edges picked by the fixed solution (this is mostly relevant for our coreset for the vertex cover problem as we explain further below).

The following proposition captures some of the interesting applications of randomized composable coresets. The proof is straightforward extension of the approach outlined in Section 1.1.4 for using composable coresets to design algorithms in these models.

Proposition 4.1.1. *Suppose ALG outputs an α-approximation randomized coreset of size s for a problem P. Let $G(V,E)$ be a graph with $m = |E|$ edges. Then, ALG implies:*

1. *A parallel algorithm in the MPC model that w.h.p. outputs an α-approximation to $P(G)$ in two rounds with $O(\sqrt{m/s})$ machines, each with $O(\sqrt{ms}+n)$ memory.*
2. *A streaming algorithm that on* random arrival *streams outputs an α-approximation to $P(G)$ with high probability using $O(\sqrt{ms})$ space.*
3. *A one-round distributed protocol that on* randomly partitioned *inputs computes an α-approximation to $P(G)$ w.h.p. using $O(s)$ communication per machine.*

4.2. Our Results and Techniques

We present our results in two parts. In the first part, we present simple constructions of randomized composable coresets that achieve $O(1)$-approximation to matching and $O(\log n)$-approximation to vertex cover using only $\widetilde{O}(n)$ space. These results already demonstrate the power of this approach over the impossibility results known for previous techniques such as linear sketches composable coresets. We also prove that the size of these coresets are optimal up to lower order terms. While our results in the this part shows a significant gap between the power of randomized composable coresets compared to previous approaches on adversarial partitions, the applications of these results to the computational models studied in this thesis (using Proposition 4.1.1) are rather weak and cannot compete with the state-of-the-arts algorithms that were designed specifically for each model.

We remedy this situation in the second part of our results by designing improved ran-

domized composable coresets using new techniques that achieve (3/2)-approximation to matching and 2-approximation to vertex cover still in $\widetilde{O}(n)$ space. Our results in this part imply a unified approach for solving these two problems across different models and improve the state-of-the-art in the aforementioned computational models in some or all parameters involved. We shall remark that the generality of randomized composable coresets can, in principle, make the problem of designing them harder or even impossible compared to solving the problem on each specific computational model. It is therefore surprising that using this unified approach, we can design essentially a single algorithm that can improve the state-of-the-art algorithms in all these models simultaneously.

4.2.1. Part One: Simple Randomized Composable Coresets

We design efficient randomized composable coresets for matching and vertex cover.

Result 4.1. *There exist randomized coresets of size $\widetilde{O}(n)$ that w.h.p. (over the random partitioning of the input) give an $O(1)$-approximation for maximum matching, and an $O(\log n)$-approximation for minimum vertex cover.*

In sharp contrast to the above result, when the graph is *adversarially* partitioned, our results from Chapter 3 show that the best approximation ratio conceivable for these problems in $\widetilde{O}(n)$ space is only $\Theta(n^{1/3})$. We further remark that Result 4.1 can also be extended to the weighted version of the problems. Using the Crouch-Stubbs technique [110] one can extend our result to achieve a coreset for weighted matching (with a factor 2 loss in approximation and extra $O(\log n)$ term in the space). Similar ideas of "grouping by weight" of edges can also be used to extend our coreset for weighted vertex cover with an $O(\log n)$ factor loss in approximation and space; we omit the details.

The $\widetilde{O}(n)$ space bound achieved by our coresets above is considered a "sweet spot" for graph streaming algorithms [254, 139] as many fundamental problems are provably intractable in $o(n)$ space (sometimes not enough to even store the answer) while admit efficient solutions in $\widetilde{O}(n)$ space. However, in the simultaneous model, these considerations imply only that the total size of all k coresets must be $\Omega(n)$, leaving open the possibility that coreset output by each machine may be as small as $\widetilde{O}(n/k)$ in size. Our next result rules out this possibility and proves the optimality of our coresets size.

Result 4.2. *Any α-approximation randomized coreset for the matching problem must have size $\Omega(n/\alpha^2)$, and any α-approximation randomized coreset for the vertex cover problem must have size $\Omega(n/\alpha)$.*

We note that our Result 4.2 can be strengthened significantly to rule out *any* summary (not necessarily a coreset) of size $o(n/\alpha^2)$ and $o(n/\alpha)$ for α-approximation of matching and vertex cover, respectively (by proving a communication complexity lower bound in

the simultaneous communication model on random partitions). However, as this extension requires a significant detour from our approach in this chapter, we omit the details here and instead refer the interested reader to the full version of our paper [30].

Our Techniques in the First Part

Randomized Coreset for Matching. Greedy and Local search algorithms are the typical choices for composable coresets (see, e.g., [186, 249]). It is then natural to consider the greedy algorithm for the maximum matching problem as a randomized coreset: the one that computes a *maximal matching*. However, one can easily show that this choice of coreset performs poorly in general; there are simple instances in which choosing arbitrary maximal matching in the graph $G^{(i)}$ results only in an $\Omega(k)$-approximation.

Somewhat surprisingly, we show that a simple change in strategy results in an efficient randomized coreset: *any maximum matching* of the graph $G^{(i)}$ can be used as an $O(1)$-approximate randomized coreset for the maximum matching problem. Unlike the previous work in [249, 186] that relied on analyzing a specific algorithm (or a specific family of algorithms) for constructing a coreset, we prove this result by exploiting structural properties of the maximum matching (i.e., the optimal solution) directly, independent of the algorithm that computes it. As a consequence, our coreset construction requires no prior coordination (such as consistent tie-breaking rules used in [249]) between the subgraphs and in fact one can use a different algorithm on each subgraph for computing the maximum matching required by the coreset.

Randomized Coreset for Vertex Cover. In the light of our coreset for the matching problem, one might wonder whether a minimum vertex cover of a graph can also be used as its randomized coreset. However, it is easy to show that the answer is negative here – there are simple instances (e.g., a star on k vertices) on which this leads to an $\Omega(k)$ approximation ratio. Indeed, the *feasibility constraint* in the vertex cover problem depends heavily on the input graph as a whole and not only the coreset computed by each machine, unlike the case for matching and in fact most problems that admit a composable coreset [43, 186, 249]. This suggests the necessity of using edges in the coreset to *certify* the feasibility of the answer. On the other hand, only sending edges seems too restrictive: a vertex of degree $n-1$ can safely be assumed to be in an optimal vertex cover, but to certify this, one needs to essentially communicate $\Omega(n)$ edges. This naturally motivates a slightly more general notion of coresets—the coreset contains both subsets of vertices (to be always included in the vertex cover) and edges (to guide the choice of additional vertices in the vertex cover).

To obtain a randomized coreset for vertex cover, we employ an iterative "peeling" process where we remove the vertices with the highest residual degree in each iteration (and add them to the final vertex cover) and continue until the residual graph is sufficiently

sparse, in which case we can return this subgraph as the coreset. The process itself is a modification of the algorithm by Parnas and Ron [266]; we point out that other modifications of this algorithm has also been used previously for matching and vertex cover [263, 205, 60].

However, to employ this algorithm as a coreset we need to argue that the set of vertices peeled across different machines is not too large as these vertices are added directly to the final vertex cover. The intuition behind this is that random partitioning of edges in the graph should result in vertices to have essentially the same degree across the machines and hence each machine should peel the same set of vertices in each iteration. But this intuition runs into a serious technical difficulty: the peeling process is quite sensitive to the exact degree of vertices and even slight changes in degree results in moving vertices between different iterations that potentially leads to a cascading effect. To address this, we design a *hypothetical* peeling process (which is aware of the actual minimum vertex cover in G) and show that the our actual peeling process is in fact "sandwiched" between two application of this peeling process with different degree threshold for peeling vertices. We then use this to argue that the set of all vertices peeled across the machines are always contained in the solution of the hypothetical peeling process which in turn can we show is a small set.

Lower Bounds for Randomized Coresets. Our lower bound results for randomized coresets for matching are based on the following simple distribution: the input graph consists of union of two bipartite graphs, one of which is a random k-regular graph G_1 with $n/2\alpha$ vertices on each side while the other graph G_2 is a perfect matching of size $n - n/2\alpha$. Thus the input graph almost certainly contains a matching of size $n - o(n)$ and any α-approximate solution must collect $\Omega(n/\alpha)$ edges from G_2 overall i.e. $\Omega(n/\alpha k)$ edges from G_2 from each machine on average. After random partitioning, the input given to each machine is essentially a matching of size $n/2\alpha$ from G_1 and a matching of size roughly n/k from G_2. The local information at each machine is not sufficient to differentiate between edges of G_1 and G_2, and thus any coreset that aims to include $\Omega(n/\alpha k)$ edges from G_2, can not reduce the input size by more than a factor of α. Somewhat similar ideas can also be shown to work for the vertex cover problem.

4.2.2. Part Two: Improved Randomized Composable Coresets

In this part, we develop a new randomized composable coreset for matching and vertex cover with improved performance compared to our Result 4.1.

Result 4.3. *There exist randomized composable coresets of size $\widetilde{O}(n)$ that for any constant $\varepsilon > 0$, give a $(3/2+\varepsilon)$-approximation for maximum matching and a $(2+\varepsilon)$-approximation for minimum vertex cover with high probability.*

Our approach in Result 4.3 is entirely different than Result 4.1, and in particular we

go beyond the ubiquitous 2-approximation barrier for matching (in Section 4.4.1, we show that our previous approach in Result 4.1 provably cannot go below 2.). Result 4.3 yields a unified framework that improves upon the state-of-the art algorithms for matching and vertex cover across several computational models, for some or all the parameters involved. Let us exhibit the most interesting results.

Streaming model. We consider single-pass streaming algorithms for matching. Computing a 2-approximation for matching (and vertex cover) in $O(n)$ space is trivial: simply maintain a maximal matching. Going beyond this barrier has remained one of the central open questions in the graph streaming literature since the introduction of the field [139]. No $o(n^2)$-space algorithm is known for this task on adversarially ordered streams and the lower bound result by Kapralov [204] (see also [160]) proves that an $\left(\frac{e}{e-1}\right)$-approximation requires $n^{1+\Omega(1/\log\log n)}$ space. To make progress on this fascinating open question, Konrad *et al.* [221] suggested the study of matching in *random arrival* streams. They presented an algorithm with approximation ratio strictly better than 2, namely $2-\delta$ for $\delta \approx 0.002$, in $O(n)$ space over random streams. A direct application of our Result 4.3 improves the approximation ratio of this algorithm significantly at a cost of a larger space.

Corollary 4.4. *There exists a single-pass streaming algorithm on random arrival streams that uses $\widetilde{O}(n\sqrt{n})$ space and with high probability (over the randomness of the stream) achieves an (almost) $(3/2)$-approximation to the maximum matching problem.*

Corollary 4.4 provides the first strong evidence of a separation between random-order and adversarial-order streams for matching, as it is the first algorithm that beats the ratio of $\left(\frac{e}{e-1}\right)$, which is known to be "hard" on adversarial streams [204]. Although the lower bound of [204] does not preclude achieving the bounds of Corollary 4.4 in an adversarial order (because our space is $\widetilde{O}(n^{1.5})$ rather than $\widetilde{O}(n)$), the proof in [204] (see also [160]) suggests that achieving such bounds is ultimately connected to further understanding of Ruzsa-Szemerédi graphs, a notoriously hard problem in additive combinatorics (see Section 2.3 and [163, 147, 18]). From a different perspective, most (but not all) streaming lower bounds are proven by bounding the (per-player) communication complexity of the problem in the blackboard communication model, including the $\left(\frac{e}{e-1}\right)$ lower bound of [204]. Our algorithm in Result 4.3 can be implemented with $\widetilde{O}(n)$ (per-player) communication in this model which goes strictly below the lower bound of [204], thus establishing the first provable separation between adversarial- and random-partitioned inputs in the blackboard model for approximating matchings.

MPC model. Maximum matching and minimum vertex cover are among the most studied graph optimization problems in the MPC and other MapReduce-style computation models

[10, 225, 9, 30, 111, 54, 175]. As an application of Result 4.3, we obtain efficient MPC algorithms for matching and vertex cover in only two rounds of computation.

Corollary 4.5. *There exist MPC algorithms that with high probability achieve an (almost)* $(3/2)$*-approximation to matching and an (almost)* 2*-approximation to vertex cover in two MPC rounds and* $\widetilde{O}(\sqrt{mn}+n)$ *memory per machine*[1].

It follows from our results in Chapter 3 (in Section 3.8) that sub-quadratic memory is not possible with one MPC round, so two rounds is optimal. Furthermore, our implementation only requires one round if the input is distributed randomly in the first place; see [249] for details on when this assumption applies.

Our matching algorithm in this part outperforms the 2-approximate maximum matching algorithm of Lattanzi *et al.* [225] in terms of both the approximation ratio (3/2 vs. 2) and round complexity (2 vs. 6) within the same memory. Our result for the matching problem is particularly interesting as all other MPC algorithms [10, 9, 54] that can achieve a better than two approximation (which is also a natural barrier for matching algorithms across different models) require a large (unspecified) constant number of rounds. Achieving the optimal 2 rounds is significant in this context, since the round complexity of MPC algorithms determines the dominant cost of the computation (see, e.g. [225, 53]), and hence minimizing the number of rounds is the primary goal in this model.

Distributed communication model. Maximum matching (and to a lesser degree vertex cover) has been studied previously in the simultaneous communication model owing to many applications of this model, including in achieving round-optimal distributed algorithms [30], proving lower bounds for dynamic graph streams [220, 14, 35, 33], and applications to mechanism design [119, 19, 118]. As an application of Result 4.3, we obtain:

Corollary 4.6. *There exist one-round communication protocols on randomly partitioned inputs that achieve (almost)* $(3/2)$*-approximation to matching and (almost)* 2*-approximation to vertex cover with high probability (over the randomness of the input partitioning) with* $\widetilde{O}(n)$ *communication per machine/player.*

Our protocols achieve optimal communication complexity (up to polylog(n) factors) by the extension of Result 4.2 in [30] we mentioned earlier.

Our Techniques in the Second Part

Our Result 4.3 is based on a novel application of *edge degree constrained subgraphs (EDCS)* that were previously introduced by Bernstein and Stein [57] for maintaining large matchings

[1]The approximation factor for vertex cover degrades to 4 if one requires local computation on each machine to be polynomial time; see Remark 4.6.8.

in dynamic graphs. Previous work on EDCS [57, 58] focused on how large a matching an EDCS contains and how it can be maintained efficiently in a dynamic graph. We instead focus on the structural properties of the EDCS, and prove several new facts in this regard. In particular, we identify the EDCS as a sparse certificate for large matchings and small vertex covers which are quite robust to sampling and composition: an ideal combination for a randomized coreset.

Roughly speaking, our results prove that if we randomly partition a graph G into k pieces $G^{(1)}, \ldots, G^{(k)}$ (as in a random k-partition), compute an EDCS $H^{(i)}$ of each graph $G^{(i)}$ (as a coreset), then the graph $H := H^{(1)} \cup \ldots \cup H^{(k)}$ (i.e., the union of the coresets) is in fact an EDCS of G. We emphasize that this composability under random partitions of the EDCS is a highly non-trivial property that we prove in this paper and is entirely different from most other similar sparse certificates for large matchings such as b-matchings. At this point, we can use the results in [57, 58] to show that H contains a $(3/2)$-approximate matching of G. We further prove that an EDCS H of a graph G also contains a 2-approximate vertex cover which lead to our coreset for vertex cover using the above approach.

4.3. Preliminaries

Let $\{X_i\}_{i=1}^n$ and $\{Y_i\}_{i=1}^n$ be a sequence of random variables such that $\mathbb{E}[X_i \mid Y_1, \ldots, Y_{i-1}] = X_{i-1}$ for all i. The sequence $\{X_i\}$ is referred to as a *martingale* with respect to $\{Y_i\}$. Azuma's inequality proves a concentration bound for martingales.

Proposition 4.3.1 (Azuma's inequality)**.** *Let $\{X_i\}_{i=0}^n$ be a martingale (with respect to some random variables $\{Y_i\}_{i=0}^n$). Suppose there exists a sequence of integers $\{c_i\}_{i=1}^n$ such that $|X_i - X_{i-1}| \leq c_i$; then,* $\Pr\Big(|X_n - X_0| \geq \lambda\Big) \leq 2 \cdot \exp\Big(-\frac{\lambda^2}{\sum_{i=1}^n c_i^2}\Big)$.

4.3.1. Edge Degree Constrained Subgraph (EDCS)

We introduce edge degree constrained subgraphs (EDCS) in this section and present several of their properties which are proven in previous work. *We emphasize that all other properties of EDCS proven in the subsequent sections are new to this thesis.* An EDCS is defined formally as follows.

Definition 4.2 ([57])**.** *For any graph $G(V, E)$ and integers $\beta \geq \beta^- \geq 0$, an* edge degree constraint subgraph (EDCS) *(G, β, β^-) is a subgraph $H := (V, E_H)$ of G with the following two properties:*

(P1) For any edge $(u, v) \in E_H$: $\mathsf{deg}_H(u) + \mathsf{deg}_H(v) \leq \beta$.

(P2) For any edge $(u, v) \in E \setminus E_H$: $\mathsf{deg}_H(u) + \mathsf{deg}_H(v) \geq \beta^-$.

We sometimes abuse the notation and use H and E_H interchangeably.

In the remainder of this thesis, we use the terms "Property (P1)" and "Property (P2)" of EDCS to refer to the first and second items in Definition 4.2 above.

One can prove the existence of an EDCS(G, β, β^-) for any graph G and parameters $\beta^- < \beta$ using the results in [58] (Theorem 3.2) which in fact shows how to maintain an EDCS efficiently in dynamic graphs (a more direct proof is presented in our paper [29]).

Lemma 4.3.2. *Any graph G contains an* EDCS(G, β, β^-) *for any parameters $\beta > \beta^-$.*

It was shown in [57] (bipartite graphs) and [58] (general graphs) that for appropriate parameters an EDCS always contains an (almost) 3/2-approximate matching of G.

Lemma 4.3.3 ([57, 58])**.** *Let $G(V, E)$ be any graph and $\varepsilon < 1/2$ be a parameter. For parameters $\lambda \geq \frac{\varepsilon}{100}$, $\beta \geq 32\lambda^{-3}$, and $\beta^- \geq (1-\lambda)\cdot\beta$, in any subgraph $H :=$* EDCS(G, β, β^-)*, $\mathsf{MM}(G) \leq \left(\frac{3}{2} + \varepsilon\right) \cdot \mathsf{MM}(H)$.*

Lemma 4.3.3 implies that an EDCS of G preserves the maximum matching of G approximately. We show in Section 4.6.1 that this is also the case for the minimum vertex cover.

4.4. Simple Randomized Composable Coresets

We present our first set of randomized composable coresets for matching and vertex cover in this section, formalizing Result 4.1.

4.4.1. An $O(1)$-Approximation Randomized Coreset for Matching

The following theorem formalizes Result 4.1 for matching.

Theorem 4.7. *Any* maximum matching *of graphs $G^{(i)}(V, E^{(i)})$ is a $(3+o(1))$-approximation randomized composable coreset of size $O(n)$ for the maximum matching problem.*

In Theorem 4.7 we assume that $\mathsf{MM}(G) = \omega(k \log n)$ since otherwise we can immediately obtain a (non-randomized) composable coreset with approximation ratio one (an exact maximum matching) and size $\widetilde{O}(k^2)$ for the matching problem using the results in [101].

A crucial building block in our proof of Theorem 4.7 is a new concentration result for the size of maximum matching in edge sampled subgraphs that we prove in the next section. This result is quite general and can be of independent interest.

Concentration of Maximum Matching Size under Edge Sampling

Let $G(V, E)$ be any arbitrary graph and $p \in (0, 1)$ be a parameter (possibly depending on size of the graph G). Define $G_p^{\mathsf{E}}(V, E_p)$ as a subgraph of G obtained by sampling each edge in E independently and with probability p, i.e., an edge sampled subgraph of G. We show that $\mathsf{MM}(G_p)$ is concentrated around its expected value.

Lemma 4.4.1. *Let $G(V, E)$ be any graph, $p \in (0, 1)$ be a parameter, and $\mathbb{E}\left[\mathsf{MM}(G_p^{\mathsf{E}})\right] \leq \mu$. For any $\lambda > 0$, $\Pr\left(\left|\mathsf{MM}(G_p^{\mathsf{E}}) - \mathbb{E}\left[\mathsf{MM}(G_p^{\mathsf{E}})\right]\right| \geq \lambda\right) \leq 2 \cdot \exp\left(-\frac{\lambda^2 \cdot p}{2 \cdot \mu}\right)$.*

Proof. For simplicity, define $G_p := G_p^{\mathsf{E}}$. Let C be any minimum vertex cover in the graph G. We use *vertex exposure* martingales over vertices in C to prove this result. Fix an arbitrary ordering of vertices in C and for any $v \in C$, define $C^{<v}$ as the set of vertices in C that appear before v in this ordering. For each $v \in C$, we define a random variable $Y_v \in \{0,1\}^{V \setminus C^{<v}}$ as a vector of indicators whether a possible edge (i.e., an edge already in G) between the vertices v and $u \in V \setminus C^{<v}$ appears in G_p or not. Since C is a vertex cover of G, every edge in G is incident on some vertex of C. As a result, the graph G_p is uniquely determined by the vectors $Y_1, \ldots, Y_{|C|}$. Define a sequence of random variables $\{X_i\}_{i=1}^{|C|}$, whereby $X_i = \mathbb{E}\left[\mathsf{MM}(G_p) \mid Y_1, \ldots, Y_i\right]$. The following claim is standard.

Claim 4.4.2. *The sequence $\{X_i\}_{i=1}^{|C|}$ is a martingale with respect to the sequence $\{Y_i\}_{i=1}^{|C|}$.*

Proof. For any $i \leq |C|$,

$$\begin{aligned}
&\mathbb{E}\left[X_i \mid Y_1, \ldots, Y_{i-1}\right] \\
&\quad= \mathop{\mathbb{E}}_{(y_1,\ldots,y_{i-1})}\left[\mathbb{E}\left[\mathsf{MM}(G_p) \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}, Y_i\right] \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}\right] \\
&\quad= \mathop{\mathbb{E}}_{(y_1,\ldots,y_{i-1})}\left[\mathsf{MM}(G_p) \mid Y_1 = y_1, \ldots, Y_{i-1} = y_{i-1}\right] \\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(as we are ``averaging out'' } Y_i \text{ in the outer expectation)} \\
&\quad= \mathbb{E}\left[\mathsf{MM}(G_p) \mid Y_1, \ldots, Y_{i-1}\right] = X_{i-1}.
\end{aligned}$$

□ Claim 4.4.2

Notice that $X_0 := \mathbb{E}\left[\mathsf{MM}(G_p)\right]$ and $X_{|C|} = \mathsf{MM}(G_p)$ as fixing $Y_1, \ldots, Y_{|C|}$ uniquely determines the graph G_p. Hence, we can use Azuma's inequality to show that value of $X_{|C|}$ is close to X_0 with high probability. To do this, we need a bound on $|C|$, as well as each term $|X_i - X_{i-1}|$. Bounding each $|X_i - X_{i-1}|$ term is quite easy; the set of edges incident on the vertex i can only change the maximum matching in G_p by 1 (as i can only be matched once), and hence $|X_i - X_{i-1}| \leq 1$. In the following, we also bound the value of $|C|$.

Claim 4.4.3. $|C| \leq 2 \cdot \mu/p$.

Proof. Since size of a minimum vertex cover of a graph G is at most twice the size of its maximum matching (Fact 2.2.5), we have that $|C| \leq 2 \cdot \mathsf{MM}(G)$. It is also straightforward to verify that $p \cdot \mathsf{MM}(G) \leq \mathbb{E}\left[\mathsf{MM}(G_p)\right] \leq \mu$, since p fraction of the edges of any maximum matching of G appear in G_p in expectation; hence $|C| \leq 2 \cdot \mu/p$. □ Claim 4.4.3

We are now ready to finalize the proof. By setting $c_i = 1$ for all $i \leq |C|$, we can use Azuma's inequality (Proposition 4.3.1) with parameters λ and c_i for the martingales $\{X_i\}$, and obtain that,

$$\Pr\left(\left|\mathsf{MM}(G_p) - \mathbb{E}\left[\mathsf{MM}(G_p)\right]\right| \geq \lambda\right) = \Pr\left(\left|X_{|C|} - X_0\right| \geq \lambda\right) \leq 2 \cdot \exp\left(-\frac{\lambda^2}{\sum_{i \in C} c_i^2}\right)$$
$$= 2 \cdot \exp\left(-\frac{\lambda^2}{|C|}\right) \underset{\text{Claim 4.4.3}}{\leq} 2 \cdot \exp\left(-\frac{\lambda^2 \cdot p}{2 \cdot \mu}\right),$$

finalizing the proof. □ Lemma 4.4.1

Proof of Theorem 4.7

Let $G(V, E)$ be any arbitrary graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. Recall that the coreset algorithm simply computes a maximum matching M_i on each graph $G^{(i)}$ for $i \in [k]$; hence, we only need to show that the graph $H(V, M_1 \cup \ldots \cup M_k)$ has a large matching compared to the graph G.

Let M^* be any fixed maximum matching in G, and let $\mu := |M^*| = \mathsf{MM}(G)$. Our approach is to show that either each graph $G^{(i)}$ has a large matching already, i.e., $|M_i| \geq \mu/3$, or many edges of M^* are picked in M_i as well. In the latter case, the union of edges in M_i for $i \in [k]$ has a large intersection with M^* and hence contains a large matching.

Define $G^-(V, E^-)$ whereby $E^- := E \setminus M^*$. Let $G_i^- := G^- \cap G^{(i)}$ be the intersection of the graph $G^{(i)}$ and G^-. Finally, define μ_i^- as the *maximum matching size* in G_i^-. Using our concentration result from the previous section, we can show that,

Claim 4.4.4. *Let $\varepsilon \in (0,1)$ be a parameter. Suppose $\mu \geq 4 \cdot \varepsilon^{-2} \cdot k \log n$; then, there exists an integer $\mu^- \in [n]$ such that with probability $1 - o(1)$ (over the random k-partition), $\mu_i^- = \mu^- \pm \varepsilon \cdot \mu$ simultaneously for all $i \in [k]$.*

Proof. Let $p = 1/k$; the graph G_i^- is a subgraph of G^- obtained by picking each edge in G^- independently and with probability p. Let $\mu^- := \mathbb{E}\left[\mathsf{MM}(G_i^-)\right] \leq \mu$ (notice that the marginal distribution of G_i^- graphs for all $i \in [k]$ are identical). By setting $\lambda = \varepsilon \cdot \mu$ in Lemma 4.4.1, we have that,

$$\Pr\left(\left|\mu_i^- - \mu^-\right| \geq \lambda\right) \underset{\text{Lemma 4.4.1}}{\leq} 2 \cdot \exp\left(-\frac{\varepsilon^2 \cdot \mu^2 \cdot p}{2 \cdot \mu}\right) \leq 2 \cdot \exp\left(-2 \log n\right) \leq \frac{1}{n^2}$$

where the second inequality is by the assumption on the value of μ. Taking a union bound over all $k \leq n$ subgraphs G_i^- for $i \in [k]$ finalizes the proof. □

In the following, we condition on the event in Claim 4.4.4. We now have,

Lemma 4.4.5. *Let μ, μ^- and ε be as in Claim 4.4.4. If $\mu^- \leq \mu/3$, then $\left|\bigcup_{i=1}^{k} M_i \cap M^*\right| \geq \mu/3 - 3\varepsilon \cdot \mu$ w.p. $1 - o(1)$.*

Proof. Fix an index $i \in [k]$ and notice that conditioning on the event in Claim 4.4.4, only fixes the set of edges in G_i^-. Let M_i^- be any maximum matching in G_i^-; by definition, $\mu_i^- = \left|M_i^-\right|$. By conditioning on the event in Claim 4.4.4, we have $\mu_i^- \leq \mu^- + 2\varepsilon \cdot \mu$. It is straightforward to verify that there are at least $|M^*| - 2 \cdot \left|M_i^-\right| = \mu - \mu_i^- \geq \mu/3 - 2\varepsilon \cdot \mu$ edges e in M^* such that neither of endpoints of e are matched by M_i^-. We refer to these edges as *free* edges and use $M_f^* \subseteq M^*$ to denote them.

Note that even after conditioning on G_i^-, the edges in M^*, and consequently M_f^*, appear in the graph $G^{(i)}$ independently and with probability $1/k$. As such, using a Chernoff bound (by assumption on the value of μ), w.p. $1-1/n^2$, $\left|M_f^*\right|/k - \varepsilon \cdot \mu/k$ edges of M_f^* appear in $G^{(i)}$. Since these edges can be directly added to the matching M_i^- (as neither endpoints of them are matched in M_i^-), this implies that there exists a matching of size $\mu_i^- + \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$ in $G^{(i)}$ w.p. $1 - 1/n^2$.

Now let M_i be the maximum matching computed by the coreset algorithm; the above argument implies that $|M_i| \geq \mu_i^- + \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$. On the other hand, notice that $\left|M_i \cap G_i^-\right| \leq \mu_i^-$ as $M_i \cap G_i^-$ forms a matching in the graph G_i^- and μ_i^- denotes the maximum matching size in this graph. This means that $|M_i \cap M^*| = \left|M_i \setminus G_i^-\right| \geq \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu)$. To finalize the proof, notice that by a union bound over all k matchings M_i, we have that with probability $1-1/n$, $\left|\bigcup_{i=1}^{k} M_i \cap M^*\right| = \sum_{i=1}^{k} |M_i \cap M^*| \geq k \cdot \frac{1}{k} \cdot (\mu/3 - 3\varepsilon \cdot \mu) = \mu/3 - 3\varepsilon \cdot \mu$. This concludes the proof. □ Lemma 4.4.5

We can now easily prove Theorem 4.7.

Proof of Theorem 4.7. By our assumption that $\mu = \mathsf{MM}(G) = \omega(k \log n)$, we can take ε in Claim 4.4.4 and Lemma 4.4.5 to be some arbitrary small constant, say $\varepsilon = o(1)$. Define μ^- as in Lemma 4.4.5. If $\mu^- > \mu/3$, we are already done as by Claim 4.4.4, for any $i \in [k]$, $|M_i| \geq \mu^- - o(\mu) \geq \mu/3 - o(\mu)$ and hence the union of matchings $M_1, \ldots, M_k$ surely has a $(3 + o(1))$ approximate matching. On the other hand, if $\mu^- \leq \mu/3$, we can apply Lemma 4.4.5, and argue that $\mu/3 - o(\mu)$ edges of the matching M^* appear in the union of matchings $M_1, \ldots, M_k$, which finalizes the proof. □ Theorem 4.7

Lower Bound on the Approximation Ratio of Maximum Matching Coreset

We also show that there exists a graph for which the approximation ratio of the coreset in Theorem 4.7 is arbitrarily close to 2. This implies that we cannot improve the analysis of this coreset much further and in particular beat the approximation ratio of 2.

Lemma 4.4.6. *There exists a graph $G(V, E)$ such that for any random k-partition of G ($k \leq n^{1-\delta}$ for any constant $\delta > 0$), the maximum matching coreset in Theorem 4.7 can only find a matching of size at most $\left(\frac{1}{2} + \frac{1}{k}\right) \cdot \mathsf{MM}(G)$ with high probability.*

Proof. The vertex set of the graph G consists of four sets of vertices L_1, L_2, R_1, R_2 with $|L_1| = \frac{n}{2} + \frac{n}{k}$ and $|L_2| = |R_1| = |R_2| = \frac{n}{2}$. G is a bipartite graph with $L_1 \cup L_2$ on one side of the bipartition and $R_1 \cup R_2$ on the other side. There is a complete bipartite graph between L_1 and R_2, a perfect matching between L_2 and R_2 and a matching of size $|R_1|$ between L_1 and R_1.

It is easy to verify that there exists a matching of size $|R_1| + |R_2| = n$ in G and hence $\mathsf{MM}(G) \geq n$. Suppose we create a random k-partition $G^{(1)}, \ldots, G^{(k)}$ of G and each machine $i \in [k]$ computes an arbitrary maximum matching M_i of its input graph (i.e., compute the $\mathsf{MaxMatching}$ coreset). In the following, we argue that the maximum matching in the graph $H(V, M_1 \cup \ldots, M_k)$ is of size $(\frac{1}{2} + \frac{1}{k}) \cdot n$ with high probability, which concludes the proof.

To prove the lemma, we need the following simple claim about the maximum matching in the edge sampled subgraphs of G.

Claim 4.4.7. *Suppose $G_p^{\mathsf{E}}(V, E_p)$ is an edge sampled subgraph of G with probability $p = 1/k$; then, w.p. $1 - 1/n^5$, there exists a matching M_p in G such that:*

1. *M_p is a maximum matching in G_p, i.e., $|M_p| = \mathsf{MM}(G)$.*
2. *No edges between L_2 and R_2 belong to M_p.*

Proof. A simple application of Chernoff bound ensures that the total number of edges between L_1 and R_1 in G_p is at most $2p \cdot n = n/k$ with probability at least $1 - 1/n^{10}$. In the following, we condition on this event. Define $M_{1,1}$ as the matching consisting of the edges between L_1 and R_1 in G_p and let $L_1^- := L_1 \setminus L_1(M_{1,1})$ be the set of vertices in L_1 that are *not* incident on $M_{1,1}$.

Consider the graph between L_1^- and R_2. Note that since $|M_{1,1}| \leq n/k$, we have $\left|L_1^-\right| \geq |R_2|$. By the independence in the sampling of edges and the fact that in G, L_1^- and R_1 forms a bipartite clique, the set of edges between L_1^- and R_2 in G_p form a random bipartite graph with probability of having each edge equal to $p = 1/k = \omega(\log n)$. Using standard facts about random graphs (see, e.g., [64], Chapter 7), this implies that there exists a matching of size $|R_2|$ between L_1^- and R_2 in G_p with probability $1 - 1/n^{10}$. Let M_p be the union of this matching and $M_{1,1}$.

It is clear that M_p does not have any edges between L_2 and R_2. To see that M_p is indeed a maximum matching of G_p, notice that all vertices in $R_1 \cup R_2$ in G_p that have non-zero degree are matched by M_p and so there cannot be any larger matching in G. $\square$ Claim 4.4.7

We are now ready to finalize the proof of Lemma 4.4.6. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with probability $1/k$. We can apply Claim 4.4.7 to each graph $G^{(i)}$ and by a union bound, w.p. $1-1/n^4$, there exists a suitable maximum matching $M_p^{(i)}$ in each graph $G^{(i)}$. Since we are choosing an *arbitrary* maximum matching of $G^{(i)}$ as its coreset, we can assume that $M_p^{(i)}$ would be chosen from each graph $G^{(i)}$, i.e., $M_i = M_p^{(i)}$ for all $i \in [k]$. This implies that no edge incident to vertices in L_2 are chosen among all coresets $M_1 \cup \ldots \cup M_k$. As a result, the maximum matching in the graph $H(V, M_1 \cup \ldots \cup M_k)$ can have size at most $|L_1| = \left(\frac{1}{2} + \frac{1}{k}\right) \cdot n$, finalizing the proof as $\mathsf{MM}(G) = n$. $\square$ Lemma 4.4.6

4.4.2. An $O(\log n)$-Approximation Randomized Coreset for Vertex Cover

The following theorem formalizes Result 4.1 for vertex cover.

Theorem 4.8. *There exists an $O(\log n)$-approximation randomized composable coreset of size $O(n \log n)$ for the vertex cover problem.*

Let $G(V, E)$ be a graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partitioning of G; we propose the following coreset for computing an approximate vertex cover of G. This coreset construction is a modification of the algorithm for vertex cover first proposed by [266].

VC-Coreset($G^{(i)}$). An algorithm for computing a composable coreset of each $G^{(i)}$.

1. Let Δ be the smallest integer such that $n/(k \cdot 2^{\Delta}) \leq 4 \log n$ and define $G_1^{(i)} := G^{(i)}$.
2. For $j = 1$ to $\Delta - 1$, let:
$$V_j^{(i)} := \left\{\text{vertices of degree} \geq n/(k \cdot 2^{j+1}) \text{ in } G_j^{(i)}\right\}$$
$$G_{j+1}^{(i)} := G_j^{(i)} \setminus V_j^{(i)}.$$
3. Return $V_{\mathsf{cs}}^{(i)} := \bigcup_{j=1}^{\Delta-1} V_j^{(i)}$ as a *fixed solution* plus the graph $G_{\Delta}^{(i)}$ as the coreset.

In VC-Coreset we allow the coreset to, in addition to returning a subgraph, identify a set of vertices (i.e., $V_{\mathsf{cs}}^{(i)}$) to be added directly to the final vertex cover. In other words, to compute a vertex cover of the graph G, we compute a vertex cover of the graph $\bigcup_{i=1}^{k} G_{\Delta}^{(i)}$ and return it together with the vertices $\bigcup_{i=1}^{k} V_{\mathsf{cs}}^{(i)}$. It is easy to see that this set of vertices indeed forms a vertex cover of G: any edge in G that belongs to $G^{(i)}$ is either incident on some $V_j^{(i)}$, and hence is covered by $V_j^{(i)}$, or is present in $G_{\Delta}^{(i)}$, and hence is covered by the vertex cover of $G_{\Delta}^{(i)}$.

In the remainder of this section, we bound the approximation ratio of this coreset. To do this, we need to prove that $\left|\bigcup_{i=1}^{k} V_{\mathsf{cs}}^{(i)}\right| = O(\log n) \cdot \mathsf{VC}(G)$. The bound on the approximation

ratio then follows as the vertex cover of $\bigcup_{i=1}^{k} G_{\Delta}^{(i)}$ can be computed to within a factor of 2.

It is easy to prove (and follows from [266]) that the set of vertices $V_{\mathsf{cs}}^{(i)}$ is of size $O(\log n) \cdot \mathsf{VC}(G)$; however, using this fact directly to bound the size of $\bigcup_{i=1}^{k} V_{\mathsf{cs}}^{(i)}$ only implies an approximation ratio of $O(k \log n)$ which is far worse than our goal of achieving an $O(\log n)$-approximation. In order to obtain the $O(\log n)$ bound, we need to argue that not only each set $V_{\mathsf{cs}}^{(i)}$ is relatively small, but also that these sets are all intersecting in many vertices. In order to do so, we introduce a hypothetical algorithm (similar to VC-Coreset) on the graph G and argue that the set $V_{\mathsf{cs}}^{(i)}$ output by $\mathsf{VC\text{-}Coreset}(G^{(i)})$ is, with high probability, a subset of the output of this hypothetical algorithm. This allows us to then bound the size of the union of the sets $V_{\mathsf{cs}}^{(i)}$ for $i \in [k]$.

Let $O^{\star}$ denote the set of vertices in an arbitrary optimum vertex cover of G and $\overline{O^{\star}} := V \setminus O^{\star}$. Consider the following hypothetical process on G (defined only for analysis):

1. Let G_1 be the bipartite graph obtained from G by removing edges between vertices in $O^{\star}$.
2. For $j = 1$ to $t := \lceil \log n \rceil$, let:

$$O_j := \big\{\text{vertices in } O^{\star} \text{ of degree} \geq n/2^j \text{ in } G_j\big\}$$
$$\overline{O}_j := \big\{\text{vertices in } \overline{O^{\star}} \text{ of degree} \geq n/2^{j+2} \text{ in } G_j\big\}$$
$$G_{j+1} := G_j \setminus (O_j \cup \overline{O}_j).$$

We first prove that the sets O_j's and $\overline{O}_j$'s in this process form an $O(\log n)$ approximation of the minimum vertex cover of G and then show that $\mathsf{VC\text{-}Coreset}(G^{(i)})$ (for any $i \in [k]$) is *mimicking* this hypothetical process in a sense that the set $V_{\mathsf{cs}}^{(i)}$ is essentially *contained* in the union of the sets O_j's and $\overline{O}_j$'s.

Lemma 4.4.8. $\left|\bigcup_{j=1}^{t} O_j \cup \overline{O}_j\right| = O(\log n) \cdot \mathsf{VC}(G)$.

Proof. Fix any $j \in [t]$; we prove that $\overline{O}_j \leq 8 \cdot \mathsf{VC}(G)$. The lemma follows from this since there are at most $O(\log n)$ different sets $\overline{O}_j$ and the union of the sets O_j's is a subset of $O^{\star}$ (with size $\mathsf{VC}(G)$).

Consider the graph G_j. The max-degree of G_j is at most $n/2^{j-1}$ by the definition of the process. Since all the edges in the graph are incident on at least one vertex of $O^{\star}$, there can be at most $|O^{\star}| \cdot n/2^{j-1}$ edges between the remaining vertices in $O^{\star}$ and $\overline{O^{\star}}$ in G_j. Moreover, any vertex in $\overline{O}_j$ has degree at least $n/2^{j+2}$ by definition and hence there can be at most $\big(|O^{\star}| \cdot n/2^{j-1}\big) \,/\, \big(n/2^{j+2}\big) \leq 8\,|O^{\star}| = 8 \cdot \mathsf{VC}(G)$ vertices in $\overline{O}_j$. □

We now prove the main relation between the sets O_j's and $\overline{O}_j$'s defined above and the intermediate sets $V_j^{(i)}$'s computed by VC-Coreset$(G^{(i)})$. The following lemma is the heart of the proof (See Figure 4a for a simple illustration).

Lemma 4.4.9. *Fix an $i \in [k]$, and let $A_j = V_j^{(i)} \cap O^\star$ and $B_j = V_j^{(i)} \cap \overline{O^\star}$. With probability $1 - O(1/n)$, for any $t \in [\Delta]$:*

1. $\bigcup_{j=1}^{t} A_j \supseteq \bigcup_{j=1}^{t} O_j$.
2. $\bigcup_{j=1}^{t} B_j \subseteq \bigcup_{j=1}^{t} \overline{O}_j$.

Proof. To simplify the notation, for any $t \in [\Delta]$, we let $A_{<t} = \bigcup_{j=1}^{t-1} A_j$ and $A_{\geq t} = \bigcup_{j=t}^{\Delta} A_j$ (and similarly for B_j's, O_j's, and $\overline{O}_j$'s). We also use $N_S(v)$ to denote the neighbor-set of the vertex v in the set $S \subseteq V$.

Note that the vertex-sets of the graphs G and $G^{(i)}$ are the same and we can "project" the sets O_j's and $\overline{O}_j$'s on graph $G^{(i)}$ as well. In other words, we can say a vertex v in $G^{(i)}$ belongs to O_j iff $v \in O_j$ in the original graph G. In the following claim, we crucially use the fact that the graph $G^{(i)}$ is obtained from G by sampling each edge w.p. $1/k$ to prove that the degree of vertices across different sets O_j's (and $\overline{O}_j$'s) in $G^{(i)}$ are essentially the same as in G (up to the scaling factor of $1/k$).

Claim 4.4.10. *For any $j \in [\Delta]$:*

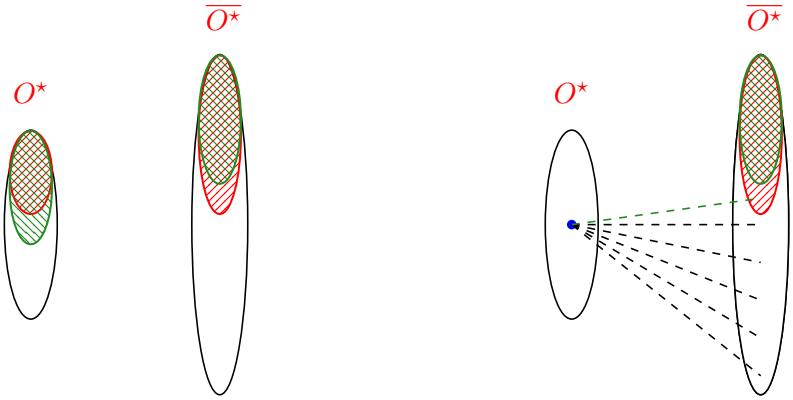
- *For any vertex $v \in O_j$, $\left|N_{\overline{O}_{\geq j}}(v)\right| \geq n/(k \cdot 2^{j+1})$ in the graph $G^{(i)}$ w.p. $1 - O(1/n^2)$.*
- *For any vertex $v \in \overline{O}_{\geq j+1}$, $\left|N_{O_{\geq j}}(v)\right| < n/(k \cdot 2^{j+1})$ in the graph $G^{(i)}$ w.p. $1 - O(1/n^2)$.*

Proof. Fix $j \in [\Delta]$ and $v \in O_j$. By definition of O_j, degree of v is at least $n/2^j$ in G_j; in other words, $\left|N_{\overline{O}^{\geq j}}(v)\right| \geq n/2^j$ in G. Since each edge in G is sampled w.p. $1/k$ in $G^{(i)}$, $\left|N_{\overline{O}^{\geq j}}(v)\right| \geq n/(k \cdot 2^j)$ in $G^{(i)}$ in expectation. By the choice of Δ, $n/(k \cdot 2^j) \geq 4 \log n$, and by Chernoff bound, w.p. $1 - O(1/n^2)$, $\left|N_{\overline{O}^{\geq j}}(v)\right| \geq n/(k \cdot 2^{j+1})$ in $G^{(i)}$.

Similarly for a vertex $v \in \overline{O}^{\geq j+1}$, degree of v is less than $n/2^{j+2}$ in G_j by definition of $\overline{O}_j$; hence, $|N_{O^{\geq j}}(v)| < n/2^{j+2}$ in the graph G. Using a similar argument as before, by Chernoff bound, w.p. $1 - O(1/n^2)$, $|N_{O^{\geq j}}(v)| < n/(k \cdot 2^{j+1})$ in $G^{(i)}$. □ Claim 4.4.10

By using a union bound on the n vertices in G, the statements in Claim 4.4.10 hold simultaneously for all vertices of G w.p. $1 - O(1/n)$; in the following we condition on this event. We now prove Lemma 4.4.9 by induction. See Figure 4b for some intuition.

Let v be a vertex that belongs to O_1; we prove that v belongs to the set $V_1^{(i)}$ of VC-Coreset, i.e., $v \in A_1$. By Claim 4.4.10 (for $j = 1$), the degree of v in $G_1^{(i)}$ is at least $n/4k$. Note that in $G_1^{(i)}$, v may also have edges to other vertices in $O^\star$ but this can only



(a) The peeled sets in the hypothetical process compared to the actual peeling process for any choice of $t \leq \Delta$.

(b) Neighborhood of a vertex in $O^\star$ can only be larger in the actual peeling process compared to the hypothetical process as green edges only appear in the actual peeling process while black edges appear in both. The opposite holds for vertices in $\overline{O^\star}$ (not drawn).

Figure 4: Illustration of Lemma 4.4.9 and its proof. Red parts correspond to the hypothetical process and green parts correspond to the actual peeling process.

increase the degree of v. This implies that v also belongs to A_1 by the threshold chosen in VC-Coreset. Similarly, let u be a vertex in $\overline{O}_{\geq 2}$ (i.e., *not* in $\overline{O}_1$); we show that u is not chosen in $V_1^{(i)}$, implying that B_1 can only contain vertices in $\overline{O}_1$. By Claim 4.4.10, degree of u in $G_1^{(i)}$ is less than $n/4k$. This implies that u does not belong to B_1. In summary, we have $O_1 \subseteq A_1$ and $B_1 \subseteq \overline{O}_1$.

Now consider some $t > 1$ and let v be a vertex in O_t. By induction, $B_{<t} \subseteq \overline{O}_{<t}$. This implies that the degree of v to $B_{\geq t}$ is at least as large as its degree to $O_{\geq t}$. Consequently, by Claim 4.4.10 (for $j = t$), degree of v in the graph $G_t^{(i)}$ is at least $n/(k \cdot 2^{t+1})$ and hence v also belongs to A_t. Similarly, fix a vertex u in $\overline{O}_{\geq t+1}$. By induction, $A_{<t} \supseteq O_{<t}$ and hence the degree of u to $A_{\geq t}$ is at most as large as its degree to $O_{\geq t}$; note that since $O^\star$ is a vertex cover, u does not have any other edge in $G_t^{(i)}$ except for the ones to $A_{\geq t}$. We can now argue as before that u does not belong to B_t. □ Lemma 4.4.9

We are now ready to prove Theorem 4.8.

Proof of Theorem 4.8. The bound on the coreset size follows immediately from the fact that the graph $G_\Delta^{(i)}$ contains at most $O(n \log n)$ edges and size of $V_{\mathsf{cs}}^{(i)}$ is at most n. As argued before, to prove the bound on the approximation ratio, we only need to show that $\bigcup_{i=1}^{k} V_{\mathsf{cs}}^{(i)}$ is of size $O(\log n) \cdot \mathsf{VC}(G)$. Let $A^{(i)} = V_{\mathsf{cs}}^{(i)} \cap O^\star$ and $B^{(i)} = V_{\mathsf{cs}}^{(i)} \cap \overline{O^\star}$; clearly, each $A^{(i)} \subseteq O^\star$ and moreover, by Lemma 4.4.9 (for $t = \Delta$), each $B^{(i)} \subseteq \cup_{j=1}^{\Delta} \overline{O}_j$. Consequently,

$\left|\bigcup_{i=1}^{k} V_{\mathsf{cs}}^{(i)}\right| \leq |O^\star| + \left|\bigcup_{j=1}^{\Delta} \overline{O}_j\right| \leq O(\log n) \cdot \mathsf{VC}(G)$ (last inequality is by Lemma 4.4.8). □

4.5. Lower Bounds for Randomized Composable Coresets

We formalize Result 4.2 in this section, proving the following two theorems.

Theorem 4.9. *For any $k = o(n/\log n)$ and $\alpha = o(\min\{n/k, k\})$, any α-approximation randomized composable coreset of the maximum matching problem is of size $\Omega(n/\alpha^2)$.*

Theorem 4.10. *For any $k = o(n/\log n)$ and $\alpha = o(\min\{n/k, k\})$, any α-approximation randomized composable coreset of the minimum vertex cover problem is of size $\Omega(n/\alpha)$.*

4.5.1. A Lower Bound for Randomized Coresets of Matching

We prove Theorem 4.9 in this section. By Yao's minimax principle (Proposition 2.7.2), to prove the lower bound in Theorem 4.9, it suffices to analyze the performance of deterministic algorithms over a fixed (hard) distribution. We propose the following distribution for this task. For simplicity of exposition, in the following, we prove a lower bound for $(\alpha/4)$-approximation algorithms; a straightforward scaling of the parameters proves the lower bound for α-approximation.

Distribution $\mathcal{D}_{\mathsf{Matching}}$. A hard input distribution for the matching problem.

- Let $G(L, R, E)$ (with $|L| = |R| = n$) be constructed as follows:
  1. Pick $A \subseteq L$ and $B \subseteq R$, each of size n/α, uniformly at random.
  2. Define E_{AB} as a set of edges between A and B, chosen by picking each edge in $A \times B$ w.p. $k \cdot \alpha/n$.
  3. Define $E_{\overline{AB}}$ as a *random* perfect matching between $\overline{A}$ and $\overline{B}$.
  4. Let $E := E_{AB} \cup E_{\overline{AB}}$.
- Let $E^{(1)}, \ldots, E^{(k)}$ be a *random k-partitioning* of E and let the input to player $P^{(i)}$ be the graph $G^{(i)}(L, R, E^{(i)})$.

Let G be a graph sampled from the distribution $\mathcal{D}_{\mathsf{Matching}}$. Notice first that the graph G always has a matching of size at least $n - n/\alpha \geq n/2$, i.e., the matching $E_{\overline{AB}}$. Additionally, it is easy to see that any matching of size more than $2n/\alpha$ in G uses at least n/α edges from $E_{\overline{AB}}$: the edges in E_{AB} can only form a matching of size n/α by construction. This implies that any $(\alpha/4)$-approximate solution requires recovering at least n/α edges from $E_{\overline{AB}}$. We prove that this is only possible if the coresets of the players are sufficiently large.

For any $i \in [k]$, define the *induced matching* $M^{(i)}$ as the unique matching in $G^{(i)}$ that is incident on *vertices of degree exactly one*, i.e., both end-points of each edge in $M^{(i)}$ have

degree one in $G^{(i)}$. We emphasize that the notion of induced matching is with respect to the entire graph and not only with respect to the vertices included in the induced matching. We have the following crucial lemma on the size of $M^{(i)}$. The proof is rather technical but standard and is hence omitted here (it can be found in our paper [30]).

Lemma 4.5.1 (cf. [30]). *W.p.* $1 - O(1/n)$, *for all* $i \in [k]$, $\left|M^{(i)}\right| = \Theta(n/\alpha)$.

Proof of Theorem 4.9. Fix any randomized composable coreset (algorithm) for the matching problem that has size $o(n/\alpha^2)$. We show that such a coreset cannot achieve a better than $(\alpha/4)$-approximation over the distribution $\mathcal{D}_{\mathsf{Matching}}$. As argued earlier, to prove this, we need to show that this coreset only contains $o(n/\alpha)$ edges from $E_{\overline{AB}}$ in expectation.

Fix any player $i \in [k]$, and let $M^{\star(i)}$ be the subset of the matching $E_{\overline{AB}}$ assigned to $P^{(i)}$. It is clear that $M^{\star(i)} \subseteq M^{(i)}$ by the definition of $M^{(i)}$. Moreover, define X_i as the random variable denoting the number of edges from $M^{\star(i)}$ that belong to the coreset sent by player $P^{(i)}$. Notice that X_i is clearly an upper bound on the number of edges of $E_{\overline{AB}}$ that are in the final matching of coordinator and also belong to the input graph of player $P^{(i)}$. In the following, we show that $\mathbb{E}[X_i] = o\left(\frac{n}{k \cdot \alpha}\right)$.

Having proved this, we have that the expected size of the output matching by the coordinator is at most $n/\alpha + \sum_{i=1}^{k} \mathbb{E}[X_i] = n/\alpha + o(n/\alpha) < (\alpha/4) \cdot \mathsf{MM}(G)$, a contradiction.

We now prove the bound on $\mathbb{E}[X_i]$. In the following, we condition on the event that $\left|M^{\star(i)}\right| = \Theta(n/k)$ and $\left|M^{(i)}\right| = \Theta(n/\alpha)$; by Chernoff bound (for the first part, since $n/k = \omega(\log n)$) and Lemma 4.5.1 (for the second part), this event happens with probability $1 - O(1/n)$. As such, this conditioning can only change $\mathbb{E}[X_i]$ by an additive factor of $O(1)$ which we ignore in the following.

A crucial property of the distribution $\mathcal{D}_{\mathsf{Matching}}$ is that the edges in $M^{\star(i)}$ and the remaining edges in $M^{(i)}$ are indistinguishable in $G^{(i)}$. More formally, for any edge $e \in G^{(i)}$,

$$\Pr\left(e \in M^{\star(i)} \mid e \in M^{(i)}\right) = \frac{\left|M^{\star(i)}\right|}{\left|M^{(i)}\right|} = \Theta(\alpha/k).$$

On the other hand, for a fixed input $M^{(i)}$ to player $P^{(i)}$, the computed coreset C_i is always the same (as the coreset is a deterministic function of the player input). Hence,

$$\mathbb{E}[X_i] = \sum_{e \in C_i} \Pr\left(e \in M_i^* \mid e \in M^{(i)}\right) = |C_i| \cdot \Theta(\alpha/k) = o(n/\alpha^2) \cdot \Theta(\alpha/k) = o\left(n/(\alpha \cdot k)\right),$$

where the second last equality is by the assumption that the size of the coreset, i.e., $|C_i|$, is $o(n/\alpha^2)$. This finalizes the proof. □

4.5.2. A Lower Bound for Randomized Coresets of Vertex Cover

In this section, we prove Theorem 4.10. By Yao's minimax principle (Proposition 2.7.2), to prove the lower bound in Theorem 4.10, it suffices to analyze the performance of deterministic algorithms over a fixed (hard) distribution. We propose the following distribution for this task. For simplicity of exposition, in the following, we prove a lower bound for $(c \cdot \alpha)$-approximation algorithms (for some constant $c > 0$); a straightforward scaling of the parameters proves the lower bound for α-approximation as well.

Distribution $\mathcal{D}_{\mathsf{VC}}$. A hard input distribution for the vertex cover problem.

- Construct $G(L, R, E)$ (with $|L| = |R| = n$) as follows:
  1. Pick $A \subseteq L$ of size n/α uniformly at random.
  2. Let E_A be a set of edges chosen by picking each edge in $A \times R$ w.p. $k/2n$.
  3. Pick a single vertex V^* uniformly at random from $\overline{A}$ and let $e^\star$ be an edge incident on V^* chosen uniformly at random.
  4. Let $E := E_A \cup \{e^\star\}$.
- Let $E^{(1)}, \ldots, E^{(k)}$ be a *random k-partitioning* of E and let the input to player $P^{(i)}$ be the graph $G^{(i)}(L, R, E^{(i)})$.

For any $i \in [k]$, we define L_i^1 as the set of vertices in L with degree *exactly one* in $G^{(i)}$. We further define R_i^1 as the set of neighbors of vertices in L_i^1 (note that vertices in R_i^1 do not *not* necessarily have degree exactly one). We start by proving a simple property.

Lemma 4.5.2. *For any $i \in [k]$, $\left|L_i^1\right| = \Theta(n/\alpha)$ and $\left|R_i^1\right| = \Theta(n/\alpha)$ w.p. $1 - o(1)$.*

Proof. Fix any player $i \in [k]$ and any vertex $v \in A$. The distribution of neighborhood of v in the graph $G^{(i)}$ is as follows: pick each vertex in R w.p. $1/2n$ independently; this is because each vertex in R is chosen w.p. $k/2n$ to be a neighbor of v in G and then each of these vertices are assigned to the graph $G^{(i)}$ w.p. $1/k$ by the random k-partitioning. As such, $\Pr\left(d(v) = 1 \text{ in } G^{(i)}\right) = \binom{n}{1} \cdot \frac{1}{2n} \cdot \left(1 - \frac{1}{2n}\right)^{n-1} \approx \frac{1}{2\sqrt{e}} = \Theta(1)$.

Consequently, we have $\mathbb{E}\left[\left|L_i^1\right|\right] = |A| \cdot \Theta(1) = \Theta(n/\alpha)$ and by Chernoff bound, $\left|L_i^1\right| = \Theta(n/\alpha)$ (note that for one player V^* would also belong to O_i but that only changes the size of $|O_i|$ by one vertex).

We bound the size of R_i^1. Each vertex in L_i^1 is choosing one vertex uniformly at random from R and hence we can model this distribution by a simple balls and bins experiment (throwing $\left|L_i^1\right|$ balls into n bins, each independently and uniformly at random), and hence by a standard fact about balls and bins experiments argue that $\left|R_i^1\right| = \Theta(n/\alpha)$ w.p. $1 - o(1)$ as well (see Proposition 2.1.5 for proof of this fact about balls and bins experiments). □

Proof of Theorem 4.10. Let i be the index of the player $P^{(i)}$ that the edge $e^\star$ is given to. We argue that if the coreset sent by player $P^{(i)}$ is of size $o(n/\alpha)$, then the coordinator cannot obtain a vertex cover of size $o(n)$. As the graph G admits a vertex cover of size $(n/\alpha+1)$ (pick A and V^*), this proves the theorem.

By Lemma 4.5.2, the set of vertices in L with degree exactly one in $G^{(i)}$ and the set of their neighbors in R, i.e., the sets L_i^1 and R_i^1, are of size $\Theta(n/\alpha)$ w.p. $1-o(1)$. In the following, we condition on this event. As the algorithm used by $P^{(i)}$ to create the coreset is deterministic, given a fixed input, it always creates the same coreset. However, a crucial property of the distribution $\mathcal{D}_{\mathsf{VC}}$ is that, conditioned on a fixed assignment to L_i^1, the vertex V^* is chosen uniformly at random from L_i^1. This implies that if the coreset of player $P^{(i)}$ contains $o(n/\alpha)$ edges, then w.p. $1-o(1)$, $e^\star$ is not part of the coreset ($e^\star$ is chosen uniformly at random from the set of all edges incident on L_i^1). Similarly, if the coreset fixes $o(n/\alpha)$ vertices to be added to the final solution, w.p. $1-o(1)$, no end point of $e^\star$ is added to this fixed set (V^* is chosen uniformly at random from L_i^1 of size $\Theta(n/\alpha)$, and the other end point of $e^\star$ is chosen uniformly at random from R_i^1 of size $\Theta(n/\alpha)$). Finally, the coresets of other players are all independent of the edge $e^\star$ and hence as long as the total number of fixed vertices sent by the players is $o(n)$, w.p. $1-o(1)$, no end points of $e^\star$ are present in the fixed solution. Conditioned on these three events, w.p. $1-o(1)$, the output of the algorithm does not cover the edge $e^\star$ and hence is not a feasible vertex cover.

We remark that this argument holds even if we are allowed to add extra vertices to the final vertex cover (other than the ones fixed by the players or computed as a vertex cover of the edges in the coresets), since conditioned on $e^\star$ not being present in any coreset, the end point of this edge are chosen uniformly at random from all vertices in $L \setminus A$ and R and hence a solution of size $o(n)$ would not contain either of them w.p. $1-o(1)$. □

4.6. Improved Randomized Composable Coresets

We present our second set of randomized composable coresets for matching and vertex cover in this section, formalizing Result 4.3. Our results in this part are based on new properties of edge-degree constrained subgraphs (EDCS) introduced in Section 4.3 that we prove in this section. As such, we start by presenting these new properties of EDCS and then use them to prove Result 4.3.

4.6.1. New Properties of Edge Degree Constrained Subgraphs

We study further properties of EDCS in this section. Although EDCS was used prior to our work, *all the properties proven in this section are entirely new to this thesis* and look at the EDCS from a different vantage point.

Previous work in [57, 58] studied the EDCS from the perspective of how large of match-

ing it contains and how it can be maintained efficiently in a dynamically changing graph. In this section, we prove several new interesting structural properties of the EDCS itself. In particular, while it is easy to see that in terms of edge sets there can be many different EDCS of some fixed graph $G(V, E)$ (consider G being a complete graph), we show that the degree distributions of every EDCS (for the same parameters β and β^-) are almost the same. In other words, the degree of any vertex v is almost the same in every EDCS of $G(V, E)$. This is in sharp contrast with similar objects such as maximum matchings or b-matchings, which can vary a lot within the same graph. This semi-uniqueness renders the EDCS extremely robust under sampling and composition as we prove next in this section.

These new structural results on EDCS are the main properties that allows their use in our coresets and parallel algorithms in the rest of the thesis.

Degree Distribution Lemma

We prove the semi-uniqueness property of EDCS defined as follows.

Lemma 4.6.1 (Degree Distribution Lemma)**.** *Fix a graph $G(V, E)$ and parameters $\beta, \beta^- = (1-\lambda)\cdot\beta$ (for $\lambda < 1/100$). For any two subgraphs A and B that are* EDCS(G, β, β^-)*, and any vertex $v \in V$, $|\mathsf{deg}_A(v) - \mathsf{deg}_B(v)| = O(\log n)\cdot\lambda^{1/2}\cdot\beta$.*

In the rest of this section, we fix the parameters β, β^- and the two EDCS A and B in Lemma 4.6.1. The general strategy of the proof is as follows. We start with a set S_1 of all vertices which has the most difference in degree between A and B. By considering the two-hop neighborhood of these vertices in A and B, we show that there exists a set S_2 of vertices in V such that the difference between the degree of vertices in A and B is almost the same as vertices in S_1, while size of S_2 is a constant factor larger than S_1. We then use this argument repeatedly to construct the next set S_3 and so on, whereby each set is larger than the previous one by a constant factor, while the gap between the degree of vertices in A and B remains almost the same as the previous set. As this geometric increase in the size of sets can only happen in a "small number" of steps (otherwise we run out of vertices), we obtain that the gap between the degree of vertices in S_1 could have not been "too large" to begin with. We now formalize this argument, starting with a technical lemma which allows us to obtain each set S_i from the set S_{i-1} in the above argument.

Lemma 4.6.2. *Fix an integer $D > 2\lambda^{1/2}\cdot\beta$ and suppose $S \subseteq V$ is such that for all $v \in S$, we have $\mathsf{deg}_A(v) - \mathsf{deg}_B(v) \geq D$. Then, there exists a set of vertices $S' \supseteq S$ such that $|S'| \geq (1 + 2\lambda^{1/2})\cdot|S|$ and for all $v \in S'$, $\mathsf{deg}_A(v) - \mathsf{deg}_B(v) \geq D - 2\lambda\cdot\beta$.*

Proof. We define the following two sets T and T':

- T is the set of all neighbors of vertices in S using *only* the edges in $A \setminus B$. In other

words, $T := \{v \in V \mid \exists u \in S \wedge (u,v) \in A \setminus B\}$.

- T' is the set of all neighbors of vertices in T using *only* the edges in $B \setminus A$. In other words, $T' := \{w \in V \mid \exists v \in T \wedge (v,w) \in B \setminus A\}$.

We start by proving the following property on the degree of vertices in the sets T, T'.

Claim 4.6.3. *We have,*

- *for all* $v \in T$, $\mathsf{deg}_B(v) - \mathsf{deg}_A(v) \geq D - \lambda \cdot \beta$.
- *for all* $w \in T'$, $\mathsf{deg}_A(w) - \mathsf{deg}_B(w) \geq D - 2\lambda \cdot \beta$.

Proof. For the first part, since $v \in T$, it means that there exists an edge $(u,v) \in A \setminus B$ such that $u \in S$. Since (u,v) belongs to A, by Property (P1) of an EDCS we have $\mathsf{deg}_A(v) \leq \beta - \mathsf{deg}_A(u) \leq \beta - \mathsf{deg}_B(u) - D$. On the other hand, since (u,v) does not belong to B, by Property (P2) of an EDCS we have $\mathsf{deg}_B(v) \geq \beta - \lambda \cdot \beta - \mathsf{deg}_B(u)$, completing the proof for vertices in T.

For the second part, since $w \in T'$, it means that there exists an edge $(v,w) \in B \setminus A$ such that $v \in T$. Since (v,w) does not belong to A, by Property (P2) of an EDCS we have $\mathsf{deg}_A(w) \geq \beta - \lambda \cdot \beta - \mathsf{deg}_A(v)$. Moreover, since (u,v) belongs to B, by Property (P1) of an EDCS, we have, $\mathsf{deg}_B(w) \leq \beta - \mathsf{deg}_B(v)$. This means that $\mathsf{deg}_A(w) - \mathsf{deg}_B(w) \geq \mathsf{deg}_B(v) - \mathsf{deg}_A(v) - \lambda \cdot \beta$ which is at least $D - 2\lambda \cdot \beta$ by the first part. □ Claim 4.6.3

Notice that since $D > 2\lambda \cdot \beta$, by Claim 4.6.3, for any vertex $v \in T$, we have $\mathsf{deg}_B(v) > \mathsf{deg}_A(v)$ and hence $S \cap T = \emptyset$ (similarly, $T \cap T' = \emptyset$, but S and T' may intersect). We define the set S' in the lemma statement to be $S' := S \cup T'$. The bound on the degree of vertices in S' follows immediately from Claim 4.6.3 (recall that vertices in S already satisfy the degree requirement for the set S'). In the following, we show that $|T' \setminus S| \geq 2\lambda^{1/2} \cdot |S|$, which finalizes the proof.

Recall that $E_{A \setminus B}(S)$ and $E_{A \setminus B}(S,T)$ denote the set of edges in subgraph $A \setminus B$ incident on vertices S, and between vertices S and T, respectively. We have,

$$\begin{aligned}
\left|E_{B\setminus A}(T, T' \setminus S)\right| &= \left|E_{B\setminus A}(T)\right| - \left|E_{B\setminus A}(T,S)\right| \\
&\text{(as all the edges in } B \setminus A \text{ that are incident on } T \text{ are going to } T') \\
&\geq \left|E_{A\setminus B}(T)\right| - \left|E_{B\setminus A}(T,S)\right| \\
&\text{(as by Claim 4.6.3, the degree of vertices in } T \text{ is larger in } B \setminus A \text{ compared to } A \setminus B) \\
&\geq \left|E_{A\setminus B}(S)\right| - \left|E_{B\setminus A}(S)\right| \\
&\text{(as all edges in } A \setminus B \text{ incident on } S \text{ are also incident on } T)
\end{aligned}$$

$$\geq |S| \cdot D.$$

(by the assumption on the degree of vertices in S in subgraphs A and B)

Finally, since B is an EDCS, the maximum degree of any vertex in $T' \setminus S$ is at most β and hence there should be at least $|S| \cdot \frac{D}{\beta} \geq 2\lambda^{1/2} \cdot |S|$ vertices in $T' \setminus S$ (as $D > 2\lambda^{1/2} \cdot \beta$). □

Proof of Lemma 4.6.1. Suppose towards a contradiction that there exists a vertex $v \in V$ s.t. $D := \mathsf{deg}_A(v) - \mathsf{deg}_B(v) \geq 3\ln(n) \cdot \lambda^{1/2} \cdot \beta$ (the other case is symmetric). Let $D_0 = D$ and $S_0 = \{v\}$ and for $i = 1$ to $t := \lambda^{-1/2} \cdot (\ln(n) + 1)$: define the set S_i and integer D_i by applying Lemma 4.6.2 to S_{i-1} and D_{i-1} (i.e., $S_i = S'$ and $D_i = D_{i-1} - 2\lambda \cdot \beta$). By the lower bound on the value of D, for any $i \in [t]$, we have that $D_i \geq D - i \cdot 2\lambda \cdot \beta > 2\lambda^{1/2} \cdot \beta$, and hence we can indeed apply Lemma 4.6.2. As a result, we have,

$$|S_t| \geq \left(1 + 2\lambda^{1/2}\right) \cdot |S_{t-1}| \geq \left(1 + 2\lambda^{1/2}\right)^t \cdot |S_0| \geq \exp\left(\lambda^{1/2} \cdot t\right) > \exp(\ln(n)) = n.$$

which is a contradiction as there are only n vertices in the graph G. Consequently, we obtain that for any vertex v, $|\mathsf{deg}_A(v) - \mathsf{deg}_B(v)| = O(\log n) \cdot \lambda^{1/2} \cdot \beta$. □ Lemma 4.6.1

EDCS in Edge Sampled Subgraphs

In this section, we prove a lemma regarding the structure of different EDCS across sampled subgraphs. We show that the degree distributions of any two EDCS for two different edge sampled subgraphs of G is almost the same no matter how the two EDCS are selected or even if the choice of the two subgraphs are *not* independent.

Lemma 4.6.4 (EDCS in Edge Sampled Subgraphs)**.** *Fix any graph $G(V, E)$ and $p \in (0, 1)$. Let G_1 and G_2 be two* edge sampled subgraphs *of G with probability p (chosen* not *necessarily independently). Let H_1 and H_2 be arbitrary EDCSs of G_1 and G_2 with parameters $(\beta, (1-\lambda) \cdot \beta)$. Suppose $\beta \geq 750 \cdot \lambda^{-2} \cdot \ln(n)$, then, with probability $1 - 4/n^9$, simultaneously for all $v \in V$: $\left|\mathsf{deg}_{H_1}(v) - \mathsf{deg}_{H_2}(v)\right| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta$.*

The proof of this lemma is along the following lines. We start with an EDCS H of the original graph G with parameters (almost) $(\beta/p, \beta^-/p)$. We then consider the set of edges from H in each of the sampled subgraphs G_1 and G_2, i.e., the two subgraphs $H'_1 := G_1 \cap H$ and $H'_2 := G_2 \cap H$. We use the randomness in the process of sampling subgraphs G_1 and G_2 to prove that with high probability both H'_1 and H'_2 form an EDCS for G_1 and G_2, respectively, with parameters (β, β^-). Finally, we use our degree distribution lemma (Lemma 4.6.1) to argue that for any arbitrary EDCS H_1 (resp. H_2) of G_1 (resp. G_2), the degree distribution of H_1 (resp. H_2) is close to H'_1 (resp. H'_2). Since the degree distributions of H'_1 and H'_2 are close to each other already (as they are both sampled subgraphs of H),

this finalizes the proof.

Proof of Lemma 4.6.4. We first prove that edge sampling an EDCS results in another EDCS for the sampled subgraph.

Claim 4.6.5. *Let H be an* EDCS(G, β_H, β_H^-) *for parameters $\beta_H := (1-\frac{\lambda}{2})\cdot\frac{\beta}{p}$ and $\beta_H^- := \beta_H - 1$. Suppose $G_p := G_p^E(V, E_p)$ is an edge sampled subgraph of G and $H_p := H \cap G_p$; then, with probability $1 - 2/n^9$:*

1. *For any vertex $v \in V$, $\left|\mathsf{deg}_{H_p}(v) - p\cdot\mathsf{deg}_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta$.*
2. *H_p is an EDCS of G_p with parameters $(\beta, (1-\lambda)\cdot\beta)$.*

Proof. For any vertex $v \in V$, $\mathbb{E}\left[\mathsf{deg}_{H_p}(v)\right] = p\cdot\mathsf{deg}_H(v)$ and $\mathsf{deg}_H(v) \leq \beta_H$ by Property (P1) of EDCS H. Moreover, since each neighbor of v in H is sampled in H_p independently, by Chernoff bound (Proposition 2.1.2), we have,

$$\Pr\left(\left|\mathsf{deg}_{H_p}(v) - p\cdot\mathsf{deg}_H(v)\right| \geq \frac{\lambda}{5}\cdot\beta\right) \leq 2\cdot\exp\left(-\frac{\lambda^2\cdot\beta}{75}\right) \leq 2\cdot\exp\left(-10\ln n\right) = \frac{2}{n^{10}},$$

where the second inequality is by the lower bound on β in Lemma 4.6.4 statement. In the following, we condition on the event that:

$$\forall v \in V \quad \left|\mathsf{deg}_{H_p}(v) - p\cdot\mathsf{deg}_H(v)\right| \leq \frac{\lambda}{5}\cdot\beta. \tag{4.1}$$

This event happens with probability at least $1-2/n^9$ by above equation and a union bound on $|V| = n$ vertices. This finalizes the proof of the first part of the claim. We are now ready to prove that H_p is indeed an EDCS$(G_p, \beta, (1-\lambda)\cdot\beta)$ conditioned on this event.

Consider any edge $(u,v) \in H_p$. Since $H_p \subseteq H$, $(u,v) \in H$ as well. Hence, we have,

$$\begin{aligned}\mathsf{deg}_{H_p}(u) + \mathsf{deg}_{H_p}(v) &\underset{\text{Eq (4.1)}}{\leq} p\cdot\Big(\mathsf{deg}_H(u) + \mathsf{deg}_H(v)\Big) + \frac{2\lambda}{5}\cdot\beta \leq p\cdot\beta_H + \frac{2\lambda}{5}\cdot\beta\\ &= (1-\frac{\lambda}{2})\cdot\beta + \frac{2\lambda}{5}\cdot\beta < \beta,\end{aligned}$$

where the second inequality is by Property (P1) of EDCS H and the equality is by the choice of β_H. As a result, H_p satisfies Property (P1) of EDCS for parameter β.

Now consider an edge $(u,v) \in G_p \setminus H_p$. Since $H_p = G_p \cap H$, $(u,v) \notin H$ as well. Hence,

$$\begin{aligned}\mathsf{deg}_{H_p}(u) + \mathsf{deg}_{H_p}(v) &\underset{\text{Eq (4.1)}}{\geq} p\cdot\Big(\mathsf{deg}_H(u) + \mathsf{deg}_H(v)\Big) - \frac{2\lambda}{5}\cdot\beta \geq p\cdot\beta_H^- - \frac{2\lambda}{5}\cdot\beta\\ &= (1-\frac{\lambda}{2})\cdot\beta - p - \frac{2\lambda}{5}\cdot\beta > (1-\lambda)\cdot\beta,\end{aligned}$$

where the second inequality is by Property (P2) of EDCS H and the equality is by the choice of β_H^-. As such, H_p satisfies Property (P2) of EDCS for parameter $(1-\lambda)\cdot\beta$ and hence H_p is indeed an EDCS$(G_p, \beta, (1-\lambda)\cdot\beta)$. $\square$ Claim 4.6.5

We continue with the proof of Lemma 4.6.4. Let H be an EDCS(G, β_H, β_H^-) for the parameters β_H, β_H^- in Claim 4.6.5. The existence of H follows from Lemma 4.3.2 as $\beta_H^- < \beta_H$. Define $\widehat{H}_1 := H \cap G_1$ and $\widehat{H}_2 := H \cap G_2$. By Claim 4.6.5, $\widehat{H}_1$ (resp. $\widehat{H}_2$) is an EDCS of G_1 (resp. G_2) with parameters $(\beta, (1-\lambda)\beta)$ with probability $1 - 4/n^9$. In the following, we condition on this event.

By Lemma 4.6.1 (Degree Distribution Lemma), since both H_1 (resp. H_2) and $\widehat{H}_1$ (resp. $\widehat{H}_2$) are EDCS for G_1 (resp. G_2), the degree of vertices in both of them should be "close" to each other. Moreover, since by Claim 4.6.5 the degree of each vertex in $\widehat{H}_1$ and $\widehat{H}_2$ is close to p times its degree in H, we can argue that the vertex degrees in H_1 and H_2 are close. Formally, for any $v \in V$, we have,

$$
\begin{aligned}
&\left|\mathsf{deg}_{H_1}(v) - \mathsf{deg}_{H_2}(v)\right| \\
&\quad\leq \left|\mathsf{deg}_{H_1}(v) - \mathsf{deg}_{\widehat{H}_1}(v)\right| + \left|\mathsf{deg}_{\widehat{H}_1}(v) - \mathsf{deg}_{\widehat{H}_2}(v)\right| + \left|\mathsf{deg}_{\widehat{H}_2}(v) - \mathsf{deg}_{H_2}(v)\right| \\
&\quad\underset{\text{Lemma 4.6.1}}{\leq} O(\log n)\cdot\lambda^{1/2}\cdot\beta + \left|\mathsf{deg}_{\widehat{H}_1}(v) - p\cdot\mathsf{deg}_H(v)\right| + \left|\mathsf{deg}_{\widehat{H}_2}(v) - p\cdot\mathsf{deg}_H(v)\right| \\
&\quad\underset{\text{Claim 4.6.5}}{\leq} O(\log n)\cdot\lambda^{1/2}\cdot\beta + O(1)\cdot\lambda\cdot\beta,
\end{aligned}
$$

finalizing the proof. $\square$ Lemma 4.6.4

EDCS Preserves Approximate Vertex Covers

Recall that Lemma 4.3.3 implies that an EDCS of a graph $G(V,E)$ preserves the maximum matching of G approximately. We also show a similar result for vertex cover. The basic idea is that in addition to computing a vertex cover for the subgraph H(to cover all the edges in H), we also add to the vertex cover all vertices that have degree at least $\geq \beta^-/2$ in H, which by Property (P2) of an EDCS covers all edges in $G \setminus H$.

Lemma 4.6.6. *Let $G(V,E)$ be any graph, $\varepsilon < 1/2$ be a parameter, and $H := \text{EDCS}(G, \beta, \beta^-)$ for parameters $\beta \geq \frac{4}{\varepsilon}$ and $\beta^- \geq \beta\cdot(1-\varepsilon/4)$. Suppose V_{HIGH} is the set of vertices $v \in V$ with $\mathsf{deg}_H(v) \geq \beta^-/2$ and V_{VC} is a minimum vertex cover of H; then $V_{\text{HIGH}} \cup V_{\text{VC}}$ is a vertex cover of G with size at most $(2+\varepsilon)\cdot\mathsf{VC}(H)$ (note that $\mathsf{VC}(H) \leq \mathsf{VC}(G)$).*

Proof. We first argue that $V_{\text{HIGH}} \cup V_{\text{VC}}$ is indeed a feasible vertex cover of G. To see this, notice that any edge $e \in H$ is covered by V_{VC}, and moreover by Property (P2) of EDCS, any edge $e \in E \setminus H$ has at least one endpoint with degree at least $\beta^-/2$ in H and hence is

covered by V_{HIGH}. In the following, we bound the size of $V_{\text{HIGH}} \setminus V_{\text{VC}}$ by $(1+\varepsilon) \cdot |V_{\text{VC}}|$, which finalizes the proof as clearly $|V_{\text{VC}}| = \mathsf{VC}(H)$.

Define $S := V_{\text{HIGH}} \setminus V_{\text{VC}}$ and let $N(S)$ be the set of all neighbors of S in the EDCS H. Since S is not part of the vertex cover V_{VC} of H, we should have $N(S) \subseteq V_{\text{VC}}$ as otherwise some edges between S and $N(S)$ would not be covered by the vertex cover V_{VC}. Now, since any vertex in S has degree at least $\beta^-/2$, we should have that degree of any vertex in $N(S)$ is at most $\beta - \beta^-/2$ in order to satisfy Property (P1) of EDCS H. Let $E(S)$ denote the set of edges incident on S in H. As all vertices in S belong to V_{HIGH}, we have that $|E(S)| \geq |S| \cdot \beta^-/2$. On the other hand, as all edges incident on S are going into $N(S)$ by definition, and since degree of vertices in $N(S)$ are bounded by $\beta - \beta^-/2$, we have $|E(S)| \leq |N(S)| \cdot (\beta - \beta^-/2)$. As such, $|S| \cdot \beta^-/2 \leq |N(S)| \cdot (\beta - \beta^-/2) \leq |V_{\text{VC}}| \cdot (1+\varepsilon) \cdot \beta^-/2$, implying that $|S| \leq (1+\varepsilon) \cdot |V_{\text{VC}}|$, which finalizes the proof. □

4.6.2. EDCS as a Randomized Composable Coreset for Matching and Vertex Cover

We introduce our randomized coresets for matching and vertex cover in this section. Both of these results are achieved by computing an EDCS of the input graph (for appropriate choice of parameters) and then applying Lemmas 4.3.3 and 4.6.6.

Computing an EDCS from Random k-Partitions

Let $G(V,E)$ be any arbitrary graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. We show that if we compute an arbitrary EDCS of each graph $G^{(i)}$ (with no coordination across different graphs) and combine them together, we obtain an EDCS for the original graph G.

1. Let $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of the graph G.
2. For any $i \in [k]$, compute $C^{(i)} := \text{EDCS}(G, \beta, (1-\lambda) \cdot \beta)$ for parameters
$$\lambda = \Theta\left((\frac{\varepsilon}{\log n})^2\right) \qquad \text{and} \qquad \beta := \Theta(\lambda^{-3} \cdot \log n).$$
3. Let $C := \bigcup_{i=1}^{k} C^{(i)}$.

Lemma 4.6.7. *W.p.* $1 - 4/n^7$, *the subgraph* C *is an* $\text{EDCS}(G, \beta_C, \beta_C^-)$ *for parameters:*

$$\lambda_C := O(\log n) \cdot \lambda^{1/2} \quad , \quad \beta_C := (1+\lambda_C) \cdot k \cdot \beta \quad \text{and} \quad \beta_C^- := (1-2\lambda_C) \cdot k \cdot \beta.$$

Proof. Recall that each graph $G^{(i)}$ is an edge sampled subgraph of G with sampling probability $p = \frac{1}{k}$. By Lemma 4.6.4 for graphs $G^{(i)}$ and $G^{(j)}$ (for $i \neq j \in [k]$) and their EDCSs

$C^{(i)}$ and $C^{(j)}$, with probability $1 - 4/n^9$, for all vertices $v \in V$:

$$|\mathsf{deg}_{C^{(i)}}(v) - \mathsf{deg}_{C^{(j)}}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta = \lambda_C \cdot \beta. \tag{4.2}$$

By taking a union bound on all $\binom{k}{2} \leq n^2$ pairs of subgraphs $G^{(i)}$ and $G^{(j)}$ for $i \neq j \in [k]$, the above property holds for all $i, j \in [k]$, with probability at least $1 - 4/n^7$. In the following, we condition on this event.

We now prove that C is indeed an EDCS(G, β_C, β_C^-). First, consider an edge $(u, v) \in C$ and let $j \in [k]$ be such that $(u, v) \in C^{(j)}$ as well. We have,

$$\begin{aligned}
\mathsf{deg}_C(u) + \mathsf{deg}_C(v) &= \sum_{i=1}^{k} \mathsf{deg}_{C^{(i)}}(u) + \sum_{i=1}^{k} \mathsf{deg}_{C^{(i)}}(v) \\
&\underset{\text{Eq (4.2)}}{\leq} k \cdot (\mathsf{deg}_{C^{(j)}}(u) + \mathsf{deg}_{C^{(j)}}(v)) + k \cdot \lambda_C \cdot \beta \\
&\leq k \cdot \beta + k \cdot \lambda_C \beta = \beta_C. \\
&\qquad \text{(by Property (P1) of EDCS } C^{(j)} \text{ with parameter } \beta)
\end{aligned}$$

Hence, C satisfies Property (P1) of EDCS for parameter β_C.

Now consider an edge $(u, v) \in G \setminus C$ and let $j \in [k]$ be such that $(u, v) \in G^{(j)} \setminus C^{(j)}$ (recall that each edge in G is sent to exactly one graph $G^{(j)}$ in the random k-partition). We have,

$$\begin{aligned}
\mathsf{deg}_C(u) + \mathsf{deg}_C(v) &= \sum_{i=1}^{k} \mathsf{deg}_{C^{(i)}}(u) + \sum_{i=1}^{k} \mathsf{deg}_{C^{(i)}}(v) \\
&\underset{\text{Eq (4.2)}}{\geq} k \cdot (\mathsf{deg}_{C^{(j)}}(u) + \mathsf{deg}_{C^{(j)}}(v)) - k \cdot \lambda_C \cdot \beta \\
&\geq k \cdot (1 - \lambda) \cdot \beta - k \cdot \lambda_C \cdot \beta \geq (1 - 2\lambda_C) \cdot k \cdot \beta = \beta_C^-. \\
&\qquad \text{(by Property (P2) of EDCS } C^{(j)} \text{ with parameter } (1 - \lambda) \cdot \beta)
\end{aligned}$$

Hence, C also satisfies Property (P2) of EDCS for parameter β_C^-, finalizing the proof. □

EDCS as a Coreset for Matching and Vertex Cover

We are now ready to present our randomized coresets for matching and vertex cover using the EDCS as the coreset, formalizing Result 4.3.

Theorem 4.11. *Let $G(V, E)$ be a graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. For any $\varepsilon \in (0, 1)$, any* EDCS$(G^{(i)}, \beta, (1 - \lambda) \cdot \beta)$ *for $\lambda := \Theta\left((\frac{\varepsilon}{\log n})^2\right)$ and $\beta := \Theta(\varepsilon^{-6} \cdot \log^7 n)$ is a $(3/2 + \varepsilon)$-approximation randomized composable coreset of size $O(n \cdot \beta)$ for the maximum matching problem.*

Proof. By Lemma 4.6.7, the union of the coresets is itself an EDCS(G, β_C, β_C^-), such that $\beta_C^- = (1 - \Theta(\varepsilon)) \cdot \beta_C$. Hence, by Lemma 4.3.3, the maximum matching in this EDCS is of size $(2/3 - \varepsilon) \cdot \mathsf{MM}(G)$. The bound on the size of the coreset follows from Property (P1) of EDCS as maximum degree in the EDCS computed by each machine is at most β and hence size of each coreset is $O(n \cdot \beta) = \widetilde{O}_\varepsilon(n)$. □

To present our coreset for the vertex cover problem, we need to use the slightly relaxed definition of randomized coreset that allows for inclusion of some part of the solution directly in the coreset.

Theorem 4.12. *Let $G(V, E)$ be a graph and $G^{(1)}, \ldots, G^{(k)}$ be a random k-partition of G. For any $\varepsilon \in (0, 1)$, any* EDCS$(G^{(i)}, \beta, (1 - \lambda) \cdot \beta)$ *for $\lambda := \Theta\left((\frac{\varepsilon}{\log n})^2\right)$ and $\beta := \Theta(\varepsilon^{-6} \cdot \log^7 n)$ plus the set of vertices with degree larger than $(1 - \Theta(\varepsilon)) \cdot \beta/2$ in the EDCS (to be added directly to the final vertex cover) is a $(2 + \varepsilon)$-approximation randomized composable coreset of size $O(n \cdot \beta)$ for the minimum vertex cover problem.*

Proof. By Lemma 4.6.7, the union of the coresets is itself an EDCS(G, β_C, β_C^-) C, such that $\beta_C^- = (1 - \Theta(\varepsilon)) \cdot \beta_C$. Suppose first that instead of each coreset fixing the set of vertices to be added to the final vertex cover, we simply add all vertices with degree more than $\beta_C^-/2$ to the vertex cover and then compute a minimum vertex cover of C. In this case, by Lemma 4.6.6, the output is a $(2 + \varepsilon)$-approximation to the minimum vertex cover of G.

To complete the argument, recall that the degree of any vertex $v \in V$ is essentially the same across all machines (up to an additive term of $\varepsilon \cdot \beta$) by Lemma 4.6.4, and hence the set of vertices with degree more than $\beta_C^-/2$ would be a subset of the set of fixed vertices across all machines. Moreover, any vertex added by any machine to the final vertex cover has degree at least $(1 - \Theta(\varepsilon)) \cdot \beta_C^-/2$ and hence we can apply Lemma 4.6.6, with a slightly smaller parameter ε to argue that the returned solution is still a $(2 + \varepsilon)$-approximation. □

Remark 4.6.8. *In the proof of Theorem 4.12, we neglected the time necessary to compute a vertex cover in the union of the coresets (as is consistent with the definition of randomized coresets). In case we require this algorithm to run in polynomial time, we need to approximate the final vertex cover in the union of the coresets as opposed to recover it exactly. In particular, by picking a 2-approximation vertex cover in the union of coreset in the proof of Lemma 4.6.6, we obtain an (almost) 4-approximation to the vertex cover of G.*

Chapter 5
Massively Parallel Algorithms for Matching and Vertex Cover with Linear Memory Per Machine

In the previous chapter, we presented a new framework for designing efficient algorithms for graph problems across different computational models such as streaming, distributed communication, and the massively parallel computation (MPC) model. In this chapter, we show further modifications of this framework that is tailored to the MPC model and allows for even more efficient algorithms for maximum matching and minimum vertex cover in this model. The materials in this chapter are based on two papers [26, 29].

The first MPC algorithms for matching and vertex cover are due to Lattanzi *et al.* [225] and obtain 2-approximation in $O(1)$ rounds and $n^{1+\Omega(1)}$ space per machine. The approximation guarantee for the matching problem was further improved to $(1+\varepsilon)$ by Ahn and Guha [10, 9]. Our own algorithms from the previous chapter achieve a $(3/2)$-approximation to maximum matching and 2-approximation to minimum vertex cover in at most *two* MPC rounds and $O(n\sqrt{n})$ space per machine. However, when the space allocated to each machine is $\widetilde{O}(n)$, the performance of all these algorithms degrade to $\Omega(\log n)$ rounds.

Recently, Czumaj *et al.* [111] presented the first $O(1)$-approximation MPC algorithm for maximum matching that uses $O((\log\log n)^2)$ rounds and $O(n)$ space per-machine (even $O(n/\text{polylog}(n))$ space). Nevertheless, several open problems left open by this work:

(i) *Can we improve the round complexity of the algorithm for [111] for matching from $O((\log\log n)^2)$ rounds to $O(\log\log n)$ rounds?*
It was conjectured by Czumaj *et al.* [111] that this is indeed possible.

(ii) *Can we extend the result of [111] to the minimum vertex cover problem also?*
The algorithm of [111] was specific to the maximum matching problem and did not extend to the closely related minimum vertex cover problem due to different technical challenges.

(iii) *Can we achieve the result of [111] using a simpler algorithm and analysis?*
The algorithm of [111] was quite intricate and required a complicated analysis.

These questions were all left open by [111] (see Section 1.4 of [111]). In this chapter, we resolve all these questions in affirmative by designing considerably simpler MPC algorithms (based on randomized composable coreset technique from the previous chapter) that achieve a $(1+\varepsilon)$-approximation to maximum matching and $O(1)$-approximation to minimum vertex cover in only $O(\log\log n)$ MPC rounds on machines with $O(n/\text{polylog}(n))$ memory.

HighLights of Our Contributions

In this chapter, we will establish new round-efficient MPC Algorithms for maximum matching and minimum vertex cover, improving the state-of-the-art (Section 5.4).

5.1. Background

The early MPC algorithms for matching and vertex cover in [225, 10, 9, 30] were all at their core based on partitioning of *edges* of the graph between different machines, process each subgraph in parallel, and then use this information to repartition graph further again, and continue until the final result is computed. While these algorithms were quite efficient, namely only required $O(1)$ rounds, when the memory per-machine was $n^{1+\Omega(1)}$, these edge sampling approaches all seemed insufficient to achieve better than $O(\log n)$ round algorithms when the memory per machine became $O(n)$ (see [111] for more detail).

Czumaj *et al.* [111] made a simple (in hindsight) but crucial observation that one can bypass the limitations of these algorithm by performing a *vertex* partitioning of the graph between the machines instead of edge partitioning. They further use a so-called *round compression* approach that corresponds to compressing multiple rounds of a particular distributed algorithm (based on the Parnas-Ron algorithm [266] introduced earlier in Section 4.4.2) into smaller number of MPC rounds by maintaining a consistent state across the local algorithms computed on each subgraph using a highly non-trivial local algorithm and analysis. We note that this idea of round compression was also implicit in our $O(\log n)$-approximation randomized coreset for vertex cover in Section 4.4.2 albeit there we applied it on edge partitioned subgraphs rather than vertex partitioned subgraphs as in [111] (indeed, our approach in Section 4.4.2 can also be used on vertex sampled subgraphs with proper modifications, resulting in an exteremly simple $O(\log \log n)$-round MPC algorithms for $O(\log n)$-approximating vertex cover on machines of memory $O(n/\text{polylog}(n))$; see [26]).

5.2. Our Results and Techniques

In this chapter, we show that one can entirely bypass the use of round compression arguments (and hence the difficulties arising in maintaining a consistent local state across the machines in parallel), by extending our second set of randomized composable coresets based on edge-degree constrained subgraphs (EDCS) in Section 4.6 to vertex partitioned subgraphs as well, resulting in the following algorithm.

Result 5.1. *There exists an MPC algorithm that for any constant $\varepsilon > 0$, with high probability, gives a $(1+\varepsilon)$-approximation to maximum matching and $O(1)$-approximation to minimum vertex cover in $O(\log \log n)$ rounds using only $O(n)$ or even $O(n/\text{polylog}(n))$ memory per machine.*

Given an existing black-box reduction [234], our Result 5.1 immediately implies a $(2+\varepsilon)$-approximation algorithm for maximum *weighted* matching in the same $O(\log\log(n))$ rounds, though with the memory per machine increased to $O(n\log(n))$.

We note that Result 5.1 improves upon the results of Czumaj *et al.* [111] on several fronts: (i) we improve the round complexity of the matching algorithm to $O(\log\log n)$, resolving a conjecture of [111] in the affirmative, (ii) we obtain an $O(1)$ approximation to vertex cover, answering another open question of [111], and (iii) we achieve all these using a considerably simpler algorithm and analysis than [111], addressing yet another open question in that work.

Techniques. We use the following recursive procedure, which crucially relies upon on the robustness properties of the EDCS proved along the way of establishing EDCS as a randomized composable coreset in Result 4.3 from previous chapter: we repeatedly compute an EDCS of the underlying graph in a distributed fashion, redistribute it again amongst multiple machines, and recursively solve the problem on this EDCS to compute an $O(1)$-approximation to matching and vertex cover. We therefore limit the memory on each machine to only $O(n)$ at the cost of increasing the number of rounds from $O(1)$ to $O(\log\log n)$. Additional ideas are needed to ensure that the approximation ratio of the algorithm does not increase beyond a fixed constant as a result of repeatedly computing an EDCS of the current graph in $O(\log\log n)$ iterations, as well as working with vertex-sampled subgraphs instead of edge-sampled subgraphs. We note that interestingly our results in this part are entirely based on the properties of EDCS that are established in this thesis and do not rely on any of prior results for EDCS in [57, 58] (in particular, do not invoke Lemma 4.3.3 that proves existence of large approximate matchings in an EDCS).

5.2.1. Subsequent Work

After our results in this chapter were published on arXiv [28], Ghaffari et al. [155] presented a result very similar to our Result 5.1: their bounds are exactly the same for matching, while for vertex cover they achieve a better approximation in the same asymptotic number of rounds: $(2+\varepsilon)$-approximation vs. our $O(1)$ approximation. Techniques-wise, our approaches are entirely different: the algorithms in [155] are again based on the round compression technique of [111] and hence suffer from the main drawback of round compression approach which is the need for an intricate local algorithm and analysis to ensure consistency between machines.

More recently, Ghaffari and Uitto [157] and Onak [262] used the round compression technique to further reduce the per-machine memory of MPC algorithms for matching and vertex cover to $n^{\Omega(1)}$ albeit at a cost of increasing the number of rounds in the algorithm to $\widetilde{O}(\sqrt{\log n})$ (which is still quadratically better than the benchmark of $O(\log n)$ rounds).

Moreover, Behnezhad *et al.* [56] and Brandt *et al.* [65] studied MPC algorithms for matching on bounded arboricity graphs (such as planar graphs) and devised an MPC algorithm that uses $O((\log\log n)^2)$ rounds and only $n^{\Omega(1)}$ memory per machine. Note that all these results are incomparable to our Result 5.1 as they use a polynomially smaller memory per machine than ours but their round complexity is either exponentially worse than our algorithm or they only work for specific family of graphs (and even there, their round complexity is quadratically worse than our bounds).

5.3. Preliminaries

EDCS in Vertex Sampled Subgraphs. As pointed out earlier, in this chapter we work with vertex sampled partitions instead of edge partitions used by our coresets. As such, we need analogue of Lemma 4.6.4 on robustness of EDCS under edge sampling for vertex sampled subgraphs as well. The main difference here is that there will be a huge gap between the degree of a vertex between the two EDCS if the vertex is sampled in one subgraph but not the other one. However, we show that the degree of vertices that are sampled in both subgraphs are almost the same across the two different (and arbitrarily chosen) EDCS for the subgraphs. The proof is almost identical to Lemma 4.6.4 and is hence omitted here.

Lemma 5.3.1 (EDCS in Vertex Sampled Subgraphs)**.** *Fix any graph $G(V,E)$ and $p \in (0,1)$. Let G_1 and G_2 be two* vertex sampled subgraphs *of G with probability p (chosen* not *necessarily independently). Let H_1 and H_2 be arbitrary EDCSs of G_1 and G_2 with parameters $(\beta,(1-\lambda)\beta)$. If $\beta \geq 750\cdot\lambda^{-2}\cdot\ln(n)$, then, with probability $1-4/n^9$, simultaneously for all $v \in G_1 \cap G_2$: $\left|\mathsf{deg}_{H_1}(v)-\mathsf{deg}_{H_2}(v)\right| \leq O(\log n)\cdot\lambda^{1/2}\cdot\beta$.*

5.4. MPC Algorithms for Matching and Vertex Cover

In this section, we show that a careful adaptation of our coresets construction in Result 4.3 together with the structural results proven for EDCS in Section 4.6.1 can be used to obtain MPC algorithms with much smaller memory while increasing the number of required rounds to only $O(\log\log n)$.

Theorem 5.2. *There exists an MPC algorithm that given a graph $G(V,E)$ with high probability computes an $O(1)$ approximation to both maximum matching and minimum vertex cover of G in $O(\log\log n+\log\left(\frac{n}{s}\right))$ MPC rounds on machines of memory $s=n^{\Omega(1)}$.*

By setting $s = O(n/\text{polylog}(n))$ in Theorem 5.2, we achieve an $O(1)$-approximation algorithm to both matching and vertex cover in $O(\log\log n)$ MPC rounds on machines of memory $O(n/\text{polylog}(n))$, formalizing Result 5.1.

In the following, for the sake of clarity, we mostly focus on proving Theorem 5.2 for the

natural case when memory per machine is $s = \widetilde{O}(n)$, and postpone the proof for all range of parameter s to Section 5.4.4. The overall idea of our algorithm is as follows. Instead of the edge sampled subgraphs used by our randomized coresets, we start by picking $k = O(n)$ *vertex* sampled subgraphs of G with sampling probability roughly $1/\sqrt{n}$ and send each to a separate machine. Each machine then locally computes an EDCS of its input (with parameters $\beta = \text{polylog}(n)$ and $\beta^- \approx \beta$) with no coordination across the machines. By Lemma 5.3.1, similar to edge-sampled subgraphs, in this case also the union of the EDCSes computed on the machines is indeed an EDCS of the original graph. However, unlike the MPC algorithm obtained by our randomized coreset approach (Corollary 4.5), where the memory per machine was as large as $\Theta(n\sqrt{n})$, here we cannot collect all these smaller EDCSes on a single machine of memory $\widetilde{O}(n)$. Instead, we repartition them across the machines again (and discard remaining edges) and repeat the previous process on this new graph. The main observation is that after each step, the maximum degree of the remaining graph (i.e., the union of all EDCSes) would drop quadratically (e.g., from potentially $\Omega(n)$ to $\widetilde{O}(\sqrt{n})$ in the first step). As such, in each subsequent step, we can pick a smaller number of vertex sampled subgraphs, each with a higher sampling probability than previous step, and still each graph fits into the memory of a single machine. Repeating this process for $O(\log\log n)$ steps reduces the maximum degree of the remaining graph to $\text{polylog}(n)$. At this point, we can store the final EDCS on a single machine and solve the problem locally.

Unfortunately this approach on its own would only yield a $(3/2)^{O(\log\log n)} = \text{polylog}(n)$ approximation to matching, since by Lemma 4.3.3 each recursion onto an EDCS of the graph could introduce a (3/2)-approximation. A similar problem exists for vertex cover. In the proof of Lemma 4.6.6, computing a vertex cover of G from its EDCS H involves two steps: we add to the vertex cover all vertices with high degree in H to cover the edges in $G \setminus H$, and then we separately compute a vertex cover for the edges in H. Since H cannot fit into a single machine, the second computation is done recursively: in each round, we find an EDCS of the current graph (which is partitioned amongst many machines), add to the vertex cover all high degree vertices in this EDCS, and then recurse onto the sparser EDCS. A straightforward analysis would only lead to an $O(\log\log n)$ approximation.

We improve the approximation factor for both vertex cover and matching by showing that they can serve as witnesses to each other. Every time we add high-degree vertices to the vertex cover, we will also find a large matching incident to these vertices: we show that this can be done in $O(1)$ parallel rounds. We then argue that their sizes are always within a constant factor of each other, so both are a constant approximation for the respective problem (by Proposition 2.2.6).

The rest of this section is organized as follows. We first present our subroutine for computing the EDCS of an input graph in parallel using vertex sampled subgraphs. Next,

we present a simple randomized algorithm for finding a large matching incident on high degree vertices of an input graph. Finally, we combine these two subroutines to provide our main parallel algorithm for approximating matching and vertex cover. We refer the interested reader to our paper [29] for minor details regarding the MPC implementation of our parallel algorithm.

5.4.1. A Parallel Algorithm for EDCS

We now present our parallel algorithm for computing an EDCS via vertex sampling. In this algorithm, the edges of the input graph as well as the output EDCS will be partitioned across multiple machines. In the following, we use a slightly involved method of sampling the vertices using limited independence. This is due to technical reasons in the MPC implementation of this algorithm which we explain in Remark 5.4.1. To avoid repeating the arguments, we present our algorithm for all range of memory $s = n^{\Omega(1)}$, but encourage the reader to consider the case of $s = n$ for more intuition.

ParallelEDCS(G, Δ, s). A parallel algorithm for EDCS of a graph G with maximum degree Δ on machines of memory $\widetilde{O}(s)$.

1. Define $p = (200 \log n) \cdot \sqrt{\frac{s}{n \cdot \Delta}}$ and $k = \frac{800 \log n}{p^2}$.
2. Create k vertex sampled subgraphs $G^{(1)}, \ldots, G^{(k)}$ on k different machines as follows:
   (a) Let $\kappa := (20 \log n)$. Each vertex v in G *independently* picks a *κ-wise independent hash function* $h_v : [k] \to [1/p]$.
   (b) The graph $G^{(i)}$ is the induced subgraph of G on vertices $v \in V$ with $h_v(i) = 0$.
3. Define parameters $\lambda := (2 \cdot \log n)^{-3}$ and $\beta := 750 \cdot \lambda^{-2} \cdot \ln(n)$.
4. For $i = 1$ to k in parallel: Compute $C^{(i)} = \text{EDCS}(G^{(i)}, \beta, (1 - \lambda) \cdot \beta)$ locally on machine i.
5. Define the *multi-graph* $C(V, E_C)$ with $E_C := \bigcup_{i=1}^{k} C^{(i)}$ (allowing for multiplicities). Notice that this multi-graph is edge partitioned across the machines.

For any vertex $v \in V$, define $I(v) \subseteq k$ as the set of indices of the subgraphs that sampled vertex v. Notice that indices in $I(v)$ are κ-wise independent random variables. Additionally, it is easy to see that each graph $G^{(i)}$ is a vertex sampled subgraph of G with sampling probability p.

Remark 5.4.1. As opposed to the previous vertex sampling approach of Czumaj *et al.* [111] that resulted in a partitioning of vertices of G across different subgraphs, our way of sampling subgraphs in ParallelEDCS results in each vertex appearing in $\Theta(p \cdot k)$ different subgraphs with high probability. This is necessary for our algorithm as we need to ensure that every edge of the input graph is sampled in this process. However, this

property introduces new challenges in the MPC implementation of our algorithm as a naive implementation of this idea requires communicating $\Theta(p \cdot k)$ messages per each edge of the graph which cannot be done within the memory restrictions of the MPC model. This is the main reason that we sample these subgraphs in ParallelEDCS with limited independence as opposed to truly independently to reduce the communication per each edge to $O(\log n)$.

We first prove some simple properties of ParallelEDCS.

Proposition 5.4.2. *For $\Delta \geq \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, with probability $1 - 2/n^8$,*

1. *For any vertex $v \in V$, $|I(v)| = p \cdot k \pm \lambda \cdot p \cdot k$.*
2. *For any edge $e \in E$, there exists at least one index $i \in [k]$ such that e belongs to $G^{(i)}$.*

Proof. Fix any vertex $v \in V$. Clearly, $\mathbb{E}|I(v)| = p \cdot k$. Moreover, $|I(v)|$ is sum of zero-one κ-wise independent random variables and hence by Chernoff bound with bounded independence (Proposition 2.1.3)

$$\Pr\left(||I(v)| - \mathbb{E}|I(v)|| \geq \lambda \cdot \mathbb{E}|I(v)|\right) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right) \leq 2 \cdot \exp\left(-10 \log n\right) \leq 1/n^{10}.$$

Note that $\mathbb{E}|I(v)| = p \cdot k = \Theta\left(\sqrt{\frac{n \cdot \Delta}{s}}\right)$, $\lambda = (2 \cdot \log n)^{-3}$ and hence by the bound on Δ, we have $\lambda^2 \cdot \mathbb{E}|I(v)|/2 = \Theta(\log n) = \kappa$, and hence we can indeed apply Proposition 2.1.3 here. By a union bound on all n vertices, the first part holds w.p. $\geq 1 - 1/n^9$.

We now prove the second part. Fix an edge $e \in E$ and define the indicator random variables $X_1, \ldots, X_k$ where $X_i = 1$ iff e is contained in the graph $G^{(i)}$. Define $X := \sum_{i=1}^{k} X_i$ to denote the number of graphs the edge e belongs to. Clearly, $\mathbb{E}[X_i] = p^2$ for all $i \in [k]$ and hence $\mathbb{E}[X] = p^2 \cdot k$. Moreover, the random variables X_i's are κ-wise independent for $\kappa = 20 \log n$. Hence, by Chernoff bound with bounded independence (Proposition 2.1.3), the probability that e belongs to no graph $G^{(i)}$, i.e., $X = 0$ is at most,

$$\Pr(X = 0) \leq \Pr\left(|X - \mathbb{E}[X]| \geq \mathbb{E}[X]\right) \leq 2 \cdot \exp\left(-\frac{\kappa}{2}\right) \leq 2 \cdot \exp\left(-10 \log n\right) \leq 1/n^{10}.$$

Again, note that $\mathbb{E}|X| = p^2 \cdot k = 800 \log n = 2\kappa$ and hence we could apply Proposition 2.1.3. By a union bound on all $O(n^2)$ edges, the second part also holds w.p. at least $1 - 1/n^8$. Another union bound on this event and the event in the first part finalizes the proof. □

We now prove that the graph C defined in the last line of ParallelEDCS is also an EDCS of G with appropriate parameters. The proof is quite similar to that of Lemma 4.6.7 with some additional care to handle the difference between vertex vs edges sampled subgraphs.

Lemma 5.4.3. *For* $\Delta \geq \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, *w.p.* $1 - 5/n^7$, C *is an* EDCS(G, β_C, β_C^-) *for parameters:*

$$\lambda_C := \lambda^{1/2} \cdot \Theta(\log n) = o(1), \qquad \beta_C := p \cdot k \cdot (1 + \lambda_C) \cdot \beta, \quad \text{and} \quad \beta_C^- = p \cdot k \cdot (1 - \lambda_C) \cdot \beta.$$

Proof. Recall that each graph $G^{(i)}$ is a vertex sampled subgraph of G with sampling probability p. Hence, by Lemma 5.3.1 and a union bound, with probability $1 - 4/n^7$, for any two subgraphs $C^{(i)}$ and $C^{(j)}$ for $i, j \in [k]$, and any vertex $v \in V^{(i)} \cap V^{(j)}$, we have,

$$|\mathsf{deg}_{C^{(i)}}(v) - \mathsf{deg}_{C^{(j)}}(v)| \leq O(\log n) \cdot \lambda^{1/2} \cdot \beta. \tag{5.1}$$

In the following, we condition on the events in Eq (5.1) and Proposition 5.4.2 which happen together with probability at least $1 - 5/n^7$.

We now prove that C is indeed an EDCS(G, β_C, β_C^-) for the given parameters. First, consider an edge $(u, v) \in C$ and let $j \in [k]$ be such that $(u, v) \in C^{(j)}$ as well. We have,

$$\begin{aligned}
&\mathsf{deg}_C(u) + \mathsf{deg}_C(v) \\
&\quad = \sum_{i \in I(u)} \mathsf{deg}_{C^{(i)}}(u) + \sum_{i \in I(v)} \mathsf{deg}_{C^{(i)}}(v) \\
&\quad \underset{\text{Eq (5.1)}}{\leq} |I(u)| \cdot \mathsf{deg}_{C^{(j)}}(u) + |I(v)| \cdot \mathsf{deg}_{C^{(j)}}(v) + (|I(u)| + |I(v)|) \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\quad \underset{\text{Proposition 5.4.2}}{\leq} p \cdot k \cdot \beta + O(\lambda) \cdot p \cdot k \cdot \beta + p \cdot k \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\qquad\qquad \text{(by Property (P1) of EDCS } C^{(j)} \text{ with parameter } \beta) \\
&\quad \leq p \cdot k \cdot (1 + \lambda_C) \cdot \beta = \beta_C.
\end{aligned}$$

Hence, C satisfies Property (P1) of EDCS for parameter β_C.

Now consider an edge $(u, v) \in G \setminus C$ and let $j \in [k]$ be such that $(u, v) \in G^{(j)} \setminus C^{(j)}$ (the existence of j follows from conditioning on the event in Proposition 5.4.2). We have,

$$\begin{aligned}
&\mathsf{deg}_C(u) + \mathsf{deg}_C(v) \\
&\quad = \sum_{i \in I(u)} \mathsf{deg}_{C^{(i)}}(u) + \sum_{i \in I(v)} \mathsf{deg}_{C^{(i)}}(v) \\
&\quad \underset{\text{Eq (5.1)}}{\geq} |I(u)| \cdot \mathsf{deg}_{C^{(j)}}(u) + |I(v)| \cdot \mathsf{deg}_{C^{(j)}}(v) - (|I(u)| + |I(v)|) \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\quad \underset{\text{Proposition 5.4.2}}{\geq} p \cdot k \cdot (1 - \lambda) \cdot \beta - O(\lambda) \cdot p \cdot k \cdot \beta - p \cdot k \cdot O(\log n) \cdot \lambda^{1/2} \cdot \beta \\
&\qquad\qquad \text{(by Property (P2) of EDCS } C^{(j)} \text{ with parameter } \beta) \\
&\quad \geq p \cdot k \cdot (1 - \lambda_C) \cdot \beta = \beta_C^-.
\end{aligned}$$

Hence, C also satisfies Property (P2) of EDCS for parameter β^-_C, finalizing the proof. □

Before moving on, we prove a simple claim that ensures that the memory of $\widetilde{O}(s)$ per machine in ParallelEDCS is enough for storing subgraph $G^{(i)}$ and computing $C^{(i)}$ locally.

Claim 5.4.4. *With probability $1 - 1/n^{18}$, the total number of edges in each subgraph $G^{(i)}$ of G in $\mathsf{ParallelEDCS}(G, \Delta, s)$ is $O(s \cdot \log^2 n)$.*

Proof. Let v be a vertex in $G^{(i)}$. By the independent sampling of vertices in a vertex sampled subgraph, we have that $\mathbb{E}\left[\mathsf{deg}_{G^{(i)}}(v)\right] = p \cdot \mathsf{deg}_G(v) \leq p \cdot \Delta = \Theta(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. By Chernoff bound, with probability $1 - 1/n^{20}$, degree of v is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. We can then take a union bound on all vertices in $G^{(i)}$ and have that with probability $1 - 1/n^{19}$, the maximum degree of $G^{(i)}$ is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n)$. At the same time, the expected number of vertices sampled in $G^{(i)}$ is at most $p \cdot n = \Theta(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n)$. Another application of Chernoff bound ensures that the total number of vertices sampled in $G^{(i)}$ is $O(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n)$ with probability $1 - 1/n^{19}$. As a result, the total number of edges in $G^{(i)}$ is $O(\sqrt{\frac{s \cdot \Delta}{n}} \cdot \log n) \cdot O(\sqrt{\frac{s \cdot n}{\Delta}} \cdot \log n) = O(s \cdot \log^2 n)$ with probability at least $1 - 1/n^{18}$. □

5.4.2. Random Match Algorithm

In our main algorithm, we need a subroutine for finding a large matching incident on the set of "high" degree vertices of a given graph G which its edges are initially partitioned across many machines. In this section, we provide such an algorithm based on a simple randomized procedure that is easily implementable in constant number of MPC rounds.

$\mathsf{RandomMatch}(G, S, \Delta)$. A parallel algorithm for finding a matching M incident on given vertices S in a graph G with maximum degree Δ.

1. Sample each vertex in S with probability $1/2$ independently to obtain a set S'.
2. For each vertex in S' pick one of its incident edges to $G \setminus S'$ uniformly at random. Let E_{smpl} be the set of these edges.
3. Let M be the matching in E_{smpl} consists of all edges with unique endpoints; these are edges $(u, v) \in E_{\mathsf{smpl}}$ such that neither u nor v are incident on any other edge of E_{smpl}.

We prove that if the set S consists of high degree vertices of G, then $\mathsf{RandomMatch}(G, S, \Delta)$ finds a large matching in S. Formally,

Lemma 5.4.5. *Suppose $G(V, E)$ is a graph with maximum degree $\Delta \geq 100 \log n$ and $S \subseteq V$ is such that for all $v \in S$, $\mathsf{deg}_G(v) \geq \Delta/3$. The size of the matching $M :=$*

$\mathsf{RandomMatch}(G, S, \Delta)$ *is in expectation* $\mathbb{E}\,|M| = \Theta(|S|)$.

Proof. Fix any vertex $v \in S'$; we argue that with high probability, degree of v to vertices in $G \setminus S'$ is at least $\Delta/7$. This follows immediately as in expectation, at most half of the neighbors of v belong to S' and we can apply Chernoff bound as $\Delta \geq 100 \log n$. We apply a union bound on all vertices in S' and in the following we condition on the event that all these vertices have at least $\Delta/7$ edges to $G \setminus S'$, which happens with high probability.

By construction, any vertex in S' has degree exactly one in E_{smpl}. As such, to lower bound the size of M, we only need to lower bound the number of vertices in $G \setminus S'$ that have degree exactly one in E_{smpl}. Fix a vertex $v \in S'$ and consider the neighbor $u \in G \setminus S'$ of v in E_{smpl}. We know that u has at most $\Delta - 1$ other neighbors in S' and each of these neighbors are choosing u with probability at most $7/\Delta$ (as each of them has at least $\Delta/7$ neighbors). Hence, $\Pr(u \text{ has degree 1 in } E_{\mathsf{smpl}}) \geq \left(1 - \frac{7}{\Delta}\right)^{\Delta-1} = \Theta(1)$. As such, in expectation, $\Theta(S)$ vertices in $G \setminus S'$ also have degree exactly one in E_{smpl}, which implies $\mathbb{E}\,|M| = \Theta(|S|)$. □

5.4.3. A Parallel Algorithm for Matching and Vertex Cover

We now present our main parallel algorithm. For sake of clarity, we present and analyze our algorithm here for the case when the memory allowed per each machine is $\widetilde{O}(n)$. We then show how to easily extend this algorithm to the case when memory per machine is $O(s)$ for any choice of $s = n^{\Omega(1)}$.

$\mathsf{ParallelAlgorithm}(G, \Delta)$. A parallel algorithm for computing a vertex cover V_{ALG} and a matching M_{ALG} of a given graph G with maximum degree at most Δ.

1. If $\Delta \leq \left(400 \cdot \log^{12} n\right)$ send G to a single machine and run the following algorithm locally: Compute a maximal matching M_{ALG} in G and let V_{ALG} be the set of vertices matched by M_{ALG}. Return V_{ALG} and M_{ALG}.
2. If $\Delta > \left(400 \cdot \log^{12} n\right)$, we run the following algorithm.
3. Compute an EDCS $C := \mathsf{ParallelEDCS}(G, \Delta, n)$ in parallel. Let β_C, β_C^- be the parameters of this EDCS (as specified in Claim 5.4.6 below).
4. Define $V_{\text{HIGH}} := \left\{v \in V \mid \mathsf{deg}_C(v) \geq \beta_C^-/2\right\}$ be the set of "high" degree vertices in C.
5. Compute a matching $M_{\text{HIGH}} := \mathsf{RandomMatch}(C, V_{\text{HIGH}}, \beta_C)$.
6. Define $V^- := V \setminus \left(V_{\text{HIGH}} \cup V(M_{\text{HIGH}})\right)$ as the set of vertices that are neither high degree in C nor matched by M_{HIGH}. Let C^- be the induced subgraph of C on vertices V^- with parallel edges removed.
7. Recursively compute $(V_{\text{REC}}, M_{\text{REC}}) := \mathsf{ParallelAlgorithm}(C^-, \beta_C)$.
8. Return $V_{\text{ALG}} := V_{\text{HIGH}} \cup V(M_{\text{HIGH}}) \cup V_{\text{REC}}$ and $M_{\text{ALG}} := M_{\text{HIGH}} \cup M_{\text{REC}}$.

We start by proving some simple properties of ParallelAlgorithm. The following claim is a direct corollary of Lemma 5.4.3 by setting $s = n$.

Claim 5.4.6. *The subgraph* $C := \mathsf{ParallelEDCS}(G, \Delta, n)$ *computed in* $\mathsf{ParallelAlgorithm}(G, \Delta)$ *w.p. at least* $1 - 1/n^5$ *is an* $\text{EDCS}(G, \beta_C, \beta_C^-)$ *for parameters:*

$$\lambda_C := o(1) \qquad \beta_C := \sqrt{\Delta} \cdot O(\log^5 n) \qquad \beta_C^- := (1 - \lambda_C) \cdot \beta_C$$

Similarly, the following claim follows easily from Lemma 5.4.5.

Claim 5.4.7. *For* $C = \text{EDCS}(G, \beta_C, \beta_C^-)$*, the matching* $M_{\text{HIGH}} = \mathsf{RandomMatch}(C, V_{\text{HIGH}}, \beta_C)$ *has expected size* $\mathbb{E}\,|M_{\text{HIGH}}| = \Omega(|V_{\text{HIGH}}|)$.

Proof. With this conditioning, the maximum degree of G is at most β_C, while the degree of vertices V_{HIGH} is at least $\beta_C^-/2 \geq \beta_C/3$. Hence, we can apply Lemma 5.4.5. □

Let T be the number of recursive calls made by $\mathsf{ParallelAlgorithm}(G, \Delta)$. We refer to any $t \in [T]$ as a *step* of ParallelAlgorithm. We bound the number of steps in ParallelAlgorithm.

Claim 5.4.8. *The total number of steps made by* $\mathsf{ParallelAlgorithm}(G, \Delta)$ *is* $T = O(\log\log\Delta)$.

Proof. Define a function $F(\Delta)$ denoting the number of recursive calls made by ParallelAlgorithm with maximum degree Δ. As $\mathsf{ParallelAlgorithm}(G, \Delta)$ runs $\mathsf{ParallelAlgorithm}(C^-, \beta_C)$ for $\beta_C < \Delta^{2/3}$, we have, $F(\Delta) \leq F(\Delta^{2/3}) + 1$ for $\Delta > \left(400 \cdot \log^{12} n\right)$ and $F(\Delta) = 1$ otherwise. It is now easy to see that $F(\Delta) = O(\log\log\Delta)$, finalizing the proof. □

In each step, ParallelAlgorithm runs the subroutines ParallelEDCS and RandomMatch once. We say that a run of ParallelEDCS is *valid* in this step iff the high probability event in Claim 5.4.6 happens. Roughly speaking, this means that ParallelEDCS is valid when it returns the "correct" output. Additionally, we say that a step of ParallelAlgorithm is valid if ParallelEDCS subroutine in this step is valid. We define $\mathcal{E}$ as the event that all T steps of $\mathsf{ParallelAlgorithm}(G, \Delta)$ are valid. By Claims 5.4.6 each step of ParallelAlgorithm is valid with probability at least $1 - 1/n^5$. As there are in total $T = O(\log\log n)$ steps by Claim 5.4.8, $\mathcal{E}$ happens with probability at least $1 - 1/n^4$. We are now ready to prove the correctness of ParallelAlgorithm.

Lemma 5.4.9. *For any graph* $G(V, E)$*,* $\mathsf{ParallelAlgorithm}(G, n)$ *with constant probability outputs a matching* M_{ALG} *which is an* $O(1)$*-approximation to the maximum matching of* G *and a vertex cover* V_{ALG} *which is an* $O(1)$*-approximation to the minimum vertex cover of* G.

Proof. It is clear that the second parameter in $\mathsf{ParallelAlgorithm}(G, n)$ is an upper bound on

the maximum degree of G and hence G satisfies the requirement of ParallelAlgorithm. In the following, we condition on the event $\mathcal{E}$ which happens with high probability by the above discussion. As such, we also have that any recursive call to ParallelAlgorithm(C^-, β_C) is valid (i.e., β_C is indeed an upper bound on degree of C^-) simply because C^- is a subgraph of an EDCS and hence its maximum degree is bounded by β_C.

We first argue that V_{ALG} and M_{ALG} are respectively a feasible vertex cover and a feasible matching of G. The case for M_{ALG} is straightforward; the set of vertices matched by M_{HIGH} is disjoint from the vertices in M_{REC} as all vertices matched by M_{HIGH} are removed in C^-, and hence (by induction) $M_{\text{ALG}} = M_{\text{HIGH}} \cup M_{\text{REC}}$ is a valid matching in G. Now consider the set of vertices V_{ALG}. By conditioning on the event $\mathcal{E}$, C is indeed an EDCS(G, β_C, β_C^-). Hence, by Property (P2) of EDCS C, any edge $e \in G \setminus C$ has at least one neighbor in V_{HIGH} and is thus covered by V_{HIGH}. Additionally, as we pick $V(M_{\text{HIGH}})$ in the vertex cover, any edge incident on these vertices are also covered. This implies that $V_{\text{HIGH}} \cup V(M_{\text{HIGH}})$ plus any vertex cover of the remaining graph C^- is a feasible vertex cover of G. As V_{REC} is a feasible vertex cover of C^- by induction, we obtain that V_{ALG} is also a feasible vertex cover of G (the base case in step 1 where a maximal matching is compute locally is trivial).

We now show that sizes of M_{ALG} and V_{ALG} are within a constant factor of each other with constant probability. By Proposition 2.2.6 this implies that both are an $O(1)$-approximation to their respective problem. At each step, the set of vertices added to the V_{ALG} are of size $|V(M_{\text{HIGH}})| + |V_{\text{HIGH}}| \leq 3\,|V_{\text{HIGH}}|$ (as M_{HIGH} is incident on V_{HIGH}). The set of edges added to matching M_{ALG} are of size M_{HIGH} which is in expectation equal to $\Theta(|V_{\text{HIGH}}|)$ by Claim 5.4.7. As such, by induction and linearity of expectation, this implies that $\mathbb{E}\,|M_{\text{ALG}}| = \Theta(|V_{\text{ALG}}|)$ (the base case is again trivial). To conclude, we can apply a Markov bound (on size of $|V_{\text{ALG}}| - |M_{\text{ALG}}|$) and obtain that with constant probability $|M_{\text{ALG}}| = \Theta(|V_{\text{ALG}}|)$. □

We note that in Lemma 5.4.9, we only achieved a constant factor probability of success for ParallelAlgorithm. We can however run this algorithm in parallel $O(\log n)$ times and pick the best solution to achieve a high probability of success while still having $\widetilde{O}(n)$ memory per machine and $O(\log \log n)$ rounds.

5.4.4. Extension to Smaller Memory Requirement: Proof of Theorem 5.2

ParallelAlgorithm can be implemented in the MPC model with machines of memory $\widetilde{O}(n)$ and $O(\log \log n)$ MPC rounds. Combining this together with Lemma 5.4.9 on the correctness of ParallelAlgorithm, we immediately obtain Theorem 5.2 for the case of $s = \widetilde{O}(n)$, i.e., an MPC algorithm with $O(1)$ approximation to both matching and vertex cover in $O(\log \log n)$ rounds with $\widetilde{O}(n)$ memory per machine.

We now show how to extend our algorithm to the case when memory per machine is

$s = n^{\Omega(1)}$. For simplicity, we assume the memory per each machine is $\widetilde{O}(s)$ as opposed to $O(s)$; rescaling the parameter s with a polylog(n) factor implies the final result. Recall that there were only two places in ParallelAlgorithm that we needed $\widetilde{O}(n)$ memory per machine: in subroutine ParallelEDCS and in step 1 of the algorithm, i.e., the base case of recursion. Consequently, we only need to make the following two changes to ParallelAlgorithm(G, Δ):

1. Firstly, in ParallelAlgorithm(G, Δ), we now run the subroutine ParallelEDCS with memory per machine equal to $O(s \cdot \text{polylog}(n))$, i.e., we run ParallelEDCS(G, Δ, s).
2. Second, we change the base case of the algorithm. Whenever $\Delta < \left(\frac{n}{s}\right) \cdot \left(400 \cdot \log^{12}(n)\right)$, instead of sending all edges to a single machine and solve the problem locally, we simply use any standard $O(\log \Delta)$-round MPC algorithm for $O(1)$-approximation to matching and vertex cover that works on machines with memory $n^{\Omega(1)}$ (for example by directly simulating the distributed peeling algorithm of [263] (see also [266]).

Previously with machines of memory $\widetilde{O}(n)$, the maximum degree of underlying graph in each step of ParallelAlgorithm was (see Claim 5.4.6):

$$\Delta_1 := n, \quad \Delta_2 := \sqrt{n} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_i := n^{1/2^i} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_T := \text{polylog}(n),$$

where $T = O(\log \log n)$ denotes the number of steps in ParallelAlgorithm. By switching to machines of memory $\widetilde{O}(s)$, we instead have,

$$\Delta_1 := n, \quad \ldots \quad \Delta_i := \frac{n}{s^{1-1/2^{i-1}}} \cdot \text{polylog}(n), \quad \ldots \quad \Delta_T := \left(\frac{n}{s}\right) \cdot \text{polylog}(n),$$

before we reach the stopping condition of ParallelAlgorithm (this follows from Lemma 5.4.3 exactly as in Claim 5.4.6). After this step, we simply compute an $O(1)$-approximate matching and vertex cover directly in the remaining graph.

The analysis of the correctness of this algorithm is exactly as before. It is also straightforward to verify that this algorithm now only needs machines of memory $O(s \cdot \log^2(n))$ by choice of ParallelEDCS. Finally, the number of rounds needed by this algorithm is $O(\log \log n)$ (for reducing the maximum degree in the graph to Δ_T) plus $O(\log \Delta_T) = O(\log \left(\frac{n}{s}\right) + \log \log n)$ (for running the distributed matching and vertex cover algorithm directly when maximum degree is at most Δ_T). This concludes the proof of Theorem 5.2 for all range of parameter $s = n^{\Omega(1)}$.

5.4.5. Further Improvements

We conclude this section by showing that using standard techniques, one can improve the approximation ratio of our matching algorithm significantly. In particular,

Corollary 5.3. *There exists an MPC algorithm that given a graph G and $\varepsilon \in (0, 1)$, with*

high probability computes a $(2+\varepsilon)$*-approximation to maximum matching of* G *in* $O(\log(1/\varepsilon)\cdot \log\log n)$ *MPC rounds and a* $(1+\varepsilon)$*-approximation in* $(1/\varepsilon)^{O(1/\varepsilon)}\cdot(\log\log n)$ *MPC rounds using only* $O(n/\text{polylog}(n))$ *memory per machine.*

In the following, we prove the first part of Corollary 5.3. The second part follows from an adaptation of a reduction of McGregor [241] (see [29]).

Proof of Corollary 5.3 - Part (1). We simply run our MPC algorithm in Theorem 5.2, to compute a matching M_{ALG}, remove all vertices matched by M_{ALG} from the graph G, and repeat. Let $\alpha = O(1)$ be the approximation ratio of the algorithm in Theorem 5.2. Suppose we repeat the above process for $T := (\alpha \cdot \log(1/\varepsilon))$ steps. For any $t \in [T]$, let M_t be the matching computed so far, i.e., the union of the all the matchings in the first t applications of our α-approximation algorithm. Also let $G_{t+1} := G \setminus V(M_t)$, i.e., the graph remained after removing vertices matched by M_t. Note that M_{t+1} is an α-approximation to the maximum matching of G_{t+1}. Moreover, $\mathsf{MM}(G_{t+1}) \geq \mathsf{MM}(G) - 2|M_t|$ as each edge in M_t can only match (and hence remove) two vertices of any maximum matching of G. This implies that $|M_{t+1}| \geq |M_t| + \frac{1}{\alpha}\cdot(\mathsf{MM}(G) - 2|M_t|)$ for all $t \in [T]$. We now have,

$$
\begin{aligned}
\mathsf{MM}(G) - 2|M_T| &\leq \left(1-\frac{2}{\alpha}\right)\cdot \mathsf{MM}(G) - 2\cdot\left(1-\frac{2}{\alpha}\right)\cdot|M_{T-1}| \\
&\leq \left(1-\frac{2}{\alpha}\right)^2 (\mathsf{MM}(G) - 2\cdot|M_{T-2}|) \\
&\qquad\qquad \text{(by applying the second equation to } M_{T-1}) \\
&\leq \left(1-\frac{2}{\alpha}\right)^T \cdot \mathsf{MM}(G) \quad \text{(by recursively applying the previous equation)} \\
&\leq \exp\left(-\frac{2}{\alpha}\cdot\alpha\cdot\log(1/\varepsilon)\right)\cdot\mathsf{MM}(G) \leq \varepsilon\cdot\mathsf{MM}(G).
\end{aligned}
$$

Hence, after $T = O(\log 1/\varepsilon)$ steps, the matching computed by the above algorithm is of size $(2+\varepsilon)\cdot\mathsf{MM}(G)$, finalizing the proof. □

Chapter 6
Massively Parallel Algorithms for Connectivity on Sparse Graphs

We now switch to studying another fundamental graph problem, namely the undirected graph connectivity problem, in the massively parallel computation model. The materials in this chapter are based on [36].

A fundamental question that shrouds the emergence of massively parallel computation (MPC) platforms is how can the additional power of the MPC paradigm (more local storage and computational power) be leveraged to achieve faster algorithms compared to classical parallel models such as PRAM?

Previous research has identified the *sparse graph connectivity* problem as a major obstacle to such improvement: While classical logarithmic-round PRAM algorithms for finding connected components in any n-vertex graph have been known for more than three decades [288, 273, 154, 106, 265, 212, 173], no $o(\log n)$-round MPC algorithms are known for this task with truly sublinear in n memory per machine. This problem arises when processing massive yet sparse graphs with $O(n)$ edges, for which the interesting setting of parameters is $n^{1-\Omega(1)}$ memory per machine. It is conjectured [23, 53, 271, 277] that achieving an $o(\log n)$-round algorithm for connectivity on general sparse graphs with $n^{1-\Omega(1)}$ per-machine memory may not be possible, and this conjecture also forms the basis for multiple *conditional* hardness results on the round complexity of other MPC problems.

In this chapter, we take an *opportunistic* approach towards the sparse graph connectivity problem, by designing an algorithm with improved performance guarantees in terms of the connectivity structure of the input graph. Formally, we design an MPC algorithm that finds all connected components with *spectral gap* at least λ in a graph in $O(\log\log n + \log(1/\lambda))$ MPC rounds and $n^{\Omega(1)}$ memory per machine. While this algorithm still requires $\Omega(\log n)$ rounds in the worst-case when components are "weakly" connected (i.e., $\lambda \approx 1/n$), it achieves an *exponential* round reduction on sparse "well-connected" components (i.e., $\lambda \geq 1/\text{polylog}(n)$) using only $n^{\Omega(1)}$ memory per machine and $\widetilde{O}(n)$ total memory, and still operates in $o(\log n)$ rounds even when $\lambda = 1/n^{o(1)}$. To best of our knowledge, this is the first non-trivial (and indeed exponential) improvement in the round complexity over PRAM algorithms, for a natural class of sparse connectivity instances.

HighLights of Our Contributions

In this chapter, we will establish:

- An MPC algorithm for the connectivity problem that uses strongly sublinear memory per-machine and can find all well-connected components of a graph in exponentially smaller number of rounds than state-of-the-art (Section 6.8)
- An MPC algorithm for the connectivity problem that uses mildly sublinear memory per-machine and can find all connected components of a graph (instead of only well-connected components in our main algorithm) in exponentially smaller number of rounds that state-of-the-art (Section 6.9).
- An unconditional lower bound for MPC algorithms for the sparse connectivity problem on well-connected graphs (Section 6.10).

6.1. Background

A fundamental question in the MPC model is:

> *How can the additional power of the MPC model (more local storage and computational power) be leveraged to achieve faster algorithms compared to classical parallel models such as PRAM algorithms?*

The answer to this question turns to be highly dependent on the type of problems at hand and the setting of parameters. For graph problems—the focus of this chapter— the first improvement over PRAM algorithms was already achieved by Karloff *et al.* [213] who developed algorithms for graph connectivity and MST in $O(1)$ MPC rounds on machines with local memory $n^{1+\Omega(1)}$; here, n is the number of vertices in the graph. This is in contrast to the $\Omega(\log n)$ round needed in the standard PRAM model for these problems (see, e.g., [288, 273, 154, 106, 265, 212, 173]). Since then, numerous algorithms have been designed for various graph problems that achieve $O(1)$ round-complexity with local memory $n^{1+\Omega(1)}$ on each machine (see, e.g., [225, 223, 9, 30] and references therein).

The next set of improvements reduced the memory per machine to $O(n)$ (possibly at the cost of a slight increase in the number of rounds). For example, an $O(1)$ round algorithm for MST and connectivity using only $O(n)$ memory per machine has been proposed in [197] building on previous work in [156, 178, 235] (see also [10, 55, 228] for related results). A recent set of results—including our results presented in Chapter 5—have also achieved $O(\log\log n)$-round algorithms for different graph problems such as matching, vertex cover, and MIS in the MPC model, when the memory per machine is $O(n)$ or even $O(n/\text{polylog}(n))$ [26, 29, 111].

Alas, this progress has came to a halt at the *truly sublinear* in n regime, i.e., $n^{1-\Omega(1)}$ space per-machine. This setting of parameter is particularly relevant to *sparse* graphs with $O(n)$ edges, as in this scenario, $\Omega(n)$ memory per-machine allows to fit the entire input on a single machine, thereby trivializing the problem.

The aforementioned line of research has identified a captivating algorithmic challenge for breaking the linear-memory barrier in the MPC model: *connectivity on sparse undirected graphs*. While classic $O(\log n)$-round PRAM algorithms for connectivity in undirected graphs have been known for more than three decades (see [288, 273, 154, 212, 173] and references therein), no faster MPC algorithm with truly sublinear $n^{1-\Omega(1)}$-memory per machine is known to date (see, e.g. [213, 218, 271]).

There are several substantial reasons for the lack of progress on this fascinating problem. On one hand, $\Omega(\log n)$ rounds are known to be necessary for a *restricted* class of (routing-style) MPC algorithms [53], and in fact it has been conjectured that this logarithmic barrier may be unavoidable for any MPC algorithm [23, 53, 271, 277]. This belief led to a series of recent results that used sparse connectivity as a hardness assumption for proving *conditional* lower bounds in the MPC model for other problems (see [23, 308] and references therein). On the other hand, as we remarked in Section 1.1.3, it was observed by [277] that proving *any* $\omega(1)$ lower bound on the round complexity of this problem would imply $\mathbf{NC}^1 \subsetneq \mathbf{P}$, a major breakthrough in complexity theory which seems beyond the reach of current techniques.

6.2. Our Results and Techniques

The *spectral gap* of a graph is defined to be the second eigenvalue of the normalized Laplacian associated with this graph (see Section 6.3.1 for more details). We use spectral gap as a measure of "connectedness" of a graph and design an *opportunistic* algorithm[1] for connectivity with improved performance guarantee depending on the spectral gap of the underlying graph.

Result 6.1. *There exists an MPC algorithm that with high probability identifies all connected components of any given sparse undirected n-vertex graph $G(V, E)$ with $\widetilde{O}(n)$ edges and a lower bound of $\lambda \in (0, 1)$ on the spectral gap of its connected components.*

For constant $\delta > 0$, the algorithm can be implemented with $O(\frac{1}{\lambda^2} \cdot n^{1-\delta} \cdot \text{polylog}(n))$ machines each with $O(n^{\delta} \cdot \text{polylog}(n))$ memory, and in $O(\log\log n + \log(1/\lambda))$ MPC rounds.

Result 6.1 can be extended to the case when the algorithm is *oblivious* to the value of λ and still manages to achieve an improved performance depending on this parameter (see Section 6.8). Our result is most interesting in the case when spectral gap of (each connected component) of the graph is lower bounded by a constant or even $1/\text{polylog}(n)$, i.e., for graphs with "well-connected" components. Examples of such graphs include random graphs[2] and expanders (see also [239, 159] for real-life examples in social networks). In this

[1] We borrow the term of "opportunistic" algorithm from Farach-Colton and Thorup [133] which defined it in the context of string matching.

[2] This means that in a probabilistic sense, this setting of parameter applies to almost all graphs.

case, we obtain an algorithm with $\widetilde{O}(n)$ total memory and $n^{\Omega(1)}$ memory per machine which can identify all connected components in only $O(\log \log n)$ rounds. To our knowledge, this constitutes the first non-trivial improvement on the standard $O(\log n)$ round algorithms for connectivity in the MPC model when the memory per machine is $n^{\Omega(1)}$ for a general family of input graphs.

Nevertheless, our algorithm in Result 6.1 still manages to achieve a non-trivial improvement even when the spectral gap is as small as $1/n^{o(1)}$. Even in this case, the algorithm requires $o(\log n)$ MPC rounds (and total memory which is larger than the input size by only an $n^{o(1)}$ factor). This means that the algorithm benefits from the extra power of the MPC model (much more local computation power) compared to the classical parallel algorithms in the PRAM model which require $\Omega(\log n)$ rounds to solve connectivity (even on sparse expanders; see below).

We also prove an *unconditional* $\Omega(\log_s n)$-round lower bound for the promise problem of connectivity on *sparse expanders* on machines with memory s. This implies that the "full power" of the MPC model is indeed required to achieve our speedup, as with $s = \text{polylog}(n)$ memory, $\Omega(\log_s n) = \widetilde{\Omega}(\log n)$ rounds are needed *even on sparse expanders* (this result, as well as a lower bound for PRAM algorithms and further discussion are presented in Section 6.10). We remark that by a result of [277], our lower bound is the best possible unconditional lower bound short of proving that $\mathbf{NC}^1 \subsetneq \mathbf{P}$ which would be a major breakthrough in complexity theory.

Finally, we note that a simple application of the toolkit we develop in proving our main result in Result 6.1 also implies that one can solve the connectivity problem in only $O(\log \log n)$ MPC rounds on *any* graph (with no assumption on spectral gap, etc.) when the memory per-machine is *mildly sublinear* in n, i.e., is $O(n/\text{polylog}(n))$. Formally,

Theorem 6.2. *There exists an MPC algorithm that given any arbitrary n-vertex graph $G(V, E)$ with high probability identifies all connected components of G in $O(\log \log n + \log\left(\frac{n}{s}\right))$ MPC rounds on machines of memory $s = n^{\Omega(1)}$.*

Theorem 6.2 is reminiscent of the recent set of results mentioned in Chapter 5 on achieving similar guarantees for other graph problems such as maximum matching and minimum vertex cover in the mildly sublinear in n per-machine memory regime. This result emphasizes the *truly sublinear* in n regime, i.e., $n^{1-\Omega(1)}$ per-machine memory, as the "real" barrier to obtaining efficient algorithms for sparse connectivity with improved performance compared to PRAM algorithms.

Techniques. The first main technical ingredient of this chapter is a distributed "data structure" for performing and processing short *independent random walks* (proportional

to the *mixing time* of each component) from *all* vertices of the graph simultaneously, whose construction takes *logarithmic* number of rounds in length of the walk. While implementing random walks in distributed and parallel settings is a well-studied problem (see, e.g., [212, 173, 283, 284] and references therein), the guarantee of our algorithm in achieving *independent* random walks across all vertices in a small number of rounds and total memory, departs from previous work (independence is crucial in the context of our algorithm). Achieving this stronger guarantee requires different tools, in particular, a method for "regularizing" our graph using a parallel implementation of the *replacement product* operation (see, e.g. [276]) that we design in this thesis.

Our second main technical ingredient is a novel *leader-election* based algorithm for finding a spanning tree of a *random graph*. The key feature of this algorithm that distinguishes it from previous MPC algorithms for sparse connectivity (see, e.g., [218, 271, 213]) is that on random graphs, it provably requires only $O(\log \log n)$ MPC rounds as opposed to $\Omega(\log n)$ (we point out that [218] also analyzed their algorithms on random graphs (see Lemma 9), but even on random graphs their algorithm requires $\Theta(\log^2 n)$ rounds). Our algorithm achieves this *exponential* speedup by contracting *quadratically* larger components to a single vertex in each step, while "preserving the randomness" in the resulting contracted graph to allow for recursion. We elaborate on our techniques in the streamlined overview of our algorithm in Section 6.4.

6.2.1. Further Related Work

Finding connected components in undirected graphs has been studied extensively in the MPC model [213, 225, 2, 10, 218, 271, 105, 203], and in the closely related distributed model of Congested Clique [197, 156, 178, 235] (see, e.g., [55] for the formal connection between the two models). In particular, for the *sparse* connectivity problem, [213, 218, 271] devised algorithms that achieve $O(\log n)$ rounds using $n^{\Omega(1)}$ memory per machine and $O(n)$ total memory. In the classical PRAM model, $O(\log n)$-round algorithms have been known for connectivity for over three decades now [288, 273, 154, 212, 173].

In the truly sublinear regime of $n^{1-\Omega(1)}$ memory per-machine, $o(\log n)$-round MPC algorithm are only known for special cases. In [23], Andoni et al. developed an approximate algorithms for approximating minimum spanning tree and Earth Mover distance for geometric graphs (complete weighted graphs for points in geometric space). In [144], Fischer and Uitto presented an $O((\log \log n)^2)$ rounds MPC algorithm for the maximal independent set problem (MIS) on trees.

6.2.2. Subsequent Work

Independently and concurrently to our work, Andoni *et al.* [24] have also studied MPC algorithms for the sparse connectivity problem with the goal of achieving improved per-

formance on graphs with "better connectivity" structure by parametrizing based on the diameter of each connected component (as opposed to spectral gap in our work). They develop an algorithm with $n^{\Omega(1)}$ memory per machine and $O(\log D \cdot \log\log_{N/n}(n))$ rounds, where D is the largest diameter of any connected component and $N = \Omega(m)$ is the total memory. Our results and that of [24] are *incomprable*: while in any graph $D = O(\log n/\lambda)$, the dependence on the number of rounds in [24] is $O(\log D \log\log n)$ for the main setting of interest in sparse connectivity when the total memory is within logarithmic factors of input size (the typical requirement of the MPC model[3] [23, 53]). As such, our algorithm achieves quadratically smaller round complexity when the spectral gap is large, i.e., is $\Omega(1)$ or even $\Omega(1/\text{polylog}(n))$ (as in random graphs and graphs with moderate expansion), while [24] achieve better performance on graphs with small spectral gap but not too-large diameter (an example is two disjoint expanders connected by an edge).

Furthermore, after our work in this chapter was published on arXiv [36], several authors studied other graph problems in the MPC model with small, i.e., $n^{\Omega(1)}$, memory per-machine. In particular, Ghaffari and Uitto [157] and Onak [262] designed $\widetilde{O}(\sqrt{\log n})$ MPC round algorithms for several graph problems such as matching, vertex cover, and maximal independent set on machines of memory $n^{\Omega(1)}$. Moreover, Behnezhad *et al.* [56] and Brandt *et al.* [65] studied a different types parametrization for several graph problems (similar to our spectral gap for connectivity), this time based on arboricity of the graph, and devised MPC algorithms that use $O((\log\log n)^2)$ rounds and only $n^{\Omega(1)}$ memory per machine for these problems on bounded arboricity graphs. We should note that the sparse connectivity problem is conjectured to require $\Omega(\log n)$ rounds even on union of cycles and hence in bounded arboricity graphs [23, 308].

6.3. Preliminaries

Notation. For a graph $G(V, E)$, we say that a subset $C \subseteq V(G)$ is a *component* of G if the induced subgraph of G on C is connected. We say that a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of $V(G)$ is a component-partition iff every C_i is a component of G.

Concise range notation. For simplicity of exposition, we use the following concise notation for representing ranges: for a value x and parameter $\delta \geq 0$, we use $[\![x \pm \delta]\!]$ to denote the range $[x - \delta, x + \delta]$. We extend this notation to numerical expressions as follows: let E be a numerical expression that apart from standard operations also contains one or more applications of the binary operator $\pm$. Let E_+ be the expression obtained from E by choosing

[3] Minimizing total memory is *critical* in the sparse connectivity problem in the MPC model. After all, the straightforward algorithm that computes the transitive closure of the graph (e.g. by matrix multiplication; see [213]) achieves $O(\log D)$ rounds, subsuming both our results and [24]; however, this algorithm requires *at least* $\Omega(n^2)$ total memory and hence does not adhere to restrictions of MPC model (or any of its more relaxed variants such as [213]).

assignment of $-$ and $+$ to replace different choices of the operator $\pm$ in order to maximize E; similarly, define E_- for minimizing E. We now define $[\![E]\!] := [E_-, E_+]$. For example, $[\![(3 \pm 2)^2]\!] = [1, 25]$ and $[\![(2 \pm 1)/(4 \pm 2)]\!] = [1/6, 3/2]$.

Almost regular graphs. Let $\Delta \geq 1$ be an integer and $\varepsilon > 0$ be any parameter. We say that a graph $G(V, E)$ is $[\![(1 \pm \varepsilon)\,\Delta]\!]$-almost-regular iff degree of any vertex in $V(G)$ belongs to $[\![(1 \pm \varepsilon)\,\Delta]\!]$. We refer to ε as the *discrepancy factor* of the an almost-regular graph.

6.3.1. Spectral Gap

Let $G(V, E)$ be an undirected graph on n vertices. We use $A_{n\times n}$ to denote the adjacency matrix of G and $D_{n\times n}$ to denote the diagonal matrix of degrees of vertices in G. We further denote the *normalized Laplacian* of G by $\mathcal{L} := I - (D^{-1/2} \cdot A \cdot D^{-1/2})$. $\mathcal{L}$ is a symmetric matrix with n real eigenvalues $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n \leq 1$. Throughout the chapter, we use $\lambda_i(G)$ to refer to the i-th smallest eigenvalue λ_i of normalized Laplacian $\mathcal{L}$ of G.

The quantity $\lambda_2(G)$ is referred to as the *spectral gap* of G, and is a quantitative measure of how "well-connected" the graph G is. For example, it is well-known that $\lambda_2(G) > 0$ iff G is connected (see, e.g., [103] for a proof), and the larger $\lambda_2(G)$ is, the graph G is more "well-connected" under various notions of connectedness. For instance, cliques and expanders, two of the most well-connected graphs, have large spectral gap; see Cheeger's inequality [92] for another such connection. In this chapter, we also use $\lambda_2(G)$ as a measure of connectivity of G and design algorithms with improved performance guarantee for graphs with larger spectral gap.

6.3.2. Random Walk on Graphs

Let $G(V, E)$ be an undirected graph. Consider the random process that starts from some vertex $v \in V$, and repeatedly moves to a neighbor of the current vertex chosen uniformly at random. We refer to this process as a *random walk*. In particular, a random walk of length t corresponds to t step of the above process. We refer to the distribution of the vertex reached by a random walk of length t from a vertex of v, as the distribution of this random walk and denote it by $\mathcal{D}_{\mathsf{RW}}(v, t)$.

Define the *random walk matrix* $W := D^{-1} \cdot A$. For any vector $v_i \in V$, let $\boldsymbol{e}_v$ denote the n-dimensional vector which is zero in all coordinates except for the i-th coordinate which is one. It is easy to see that for any integer $t \geq 1$, the vector $W^t \cdot \boldsymbol{e}_v$ corresponds to the distribution of a random walk of length t starting from v, i.e., $\mathcal{D}_{\mathsf{RW}}(v, t)$. We use $\pi = \pi(G)$ to denote the *stationary distribution* of a random walk on a graph G, where for any $v \in V$, $\pi_v := \frac{d_v}{2m}$. It is immediate to verify that $W \cdot \pi = \pi$.

As random walks on arbitrary connected graphs do not necessarily converge to their stationary distribution (i.e., when the underlying graph is bipartite), we further consider

lazy random walks. In a lazy random walk of length t, starting from some vertex $v \in V$, for t steps we either stay at the current vertex with probability half, or move to a neighbor of the current vertex chosen uniformly at random. We define the *lazy random walk matrix* as $\overline{W} := (I + W)/2$ which is the transition matrix of a lazy random walk. It is easy to verify that π is also the stationary distribution for a lazy random walk.

Mixing Time. For any $\gamma > 0$, we define the γ-mixing time of G, denoted by $T_\gamma(G)$ to be the smallest integer $t \geq 1$, such that the distribution of a lazy random walk of length t on G starting from any arbitrary vertex become γ-close to the stationary distribution in total variation distance. Formally, $T_\gamma(G) := \min_{t \geq 1} \max_{v \in V(G)} \left\{ \left| \overline{W}^t \cdot \boldsymbol{e}_v - \pi \right|_{tvd} \leq \gamma \right\}$.

The following well-known proposition relates the mixing time of a graph G to its spectral gap (see, e.g. [103] chapter 1.5 for a proof).

Proposition 6.3.1. *For any connected graph $G(V, E)$ and $\gamma < 1$, $T_\gamma(G) = O\left(\frac{\log(n/\gamma)}{\lambda_2(G)}\right)$.*

6.3.3. Random Graphs

For any integers $n, d \geq 1$, we use $\mathbb{G}(n, d)$ to denote the distribution on random undirected graphs G on n vertices chosen by picking for each vertex $v \in V(G)$, $\lfloor d/2 \rfloor$ outgoing edges (u, v) for v chosen uniformly at random (with replacement) from $V(G)$ and then removing the direction of edges. Note that this notion of a random graph is related but not identical to the more familiar family of Erdos-Renyi random graphs.

Throughout the chapter we use several properties of these random graphs that we present in this section. The proofs of the following propositions are standard and follow from similar arguments in Erdos-Renyi random graphs (we refer the reader to [64]).

Proposition 6.3.2 (**Almost-regularity**). *Suppose $d \geq 4 \log n/\varepsilon^2$ for some parameter $\varepsilon \in (0, 1)$. A graph $G \sim \mathbb{G}(n, d)$ is an $[\![(1 \pm \varepsilon)\, d]\!]$-almost-regular with high probability.*

Proposition 6.3.3 (**Connectivity**). *A graph $G \sim \mathbb{G}(n, d)$ for $d \geq c \log n$ is connected with probability at least $1 - 1/n^{(c/4)}$.*

Proposition 6.3.4 (**Expansion**). *Suppose $G \sim \mathbb{G}(n, d)$ for $d \geq c \log n$. Then, with probability at least $1 - 1/n^{(c/4)}$:*

1. *For any set $S \subseteq V(G)$, the neighborset $N(S)$ of S in G has size*

$$|N(S)| \geq \min\{2n/3, d/12 \cdot |S|\}.$$

2. *The mixing time of G is $T_\gamma(G) = O(d^2 \cdot \log(n/\gamma))$ for any $\gamma < 1$.*

6.4. Technical Overview of Our Algorithm

In this section, we present a streamlined overview of our technical approach for proving Result 6.1. For simplicity, we focus here mainly on the case $\lambda = 1/\text{polylog}(n)$, i.e., the case of graphs with moderate (spectral) expansion.

The general strategy behind our algorithm is the natural and familiar approach of *improving the connectivity* of the underlying graph before finding its connected components (see, e.g., the celebrated log-space connectivity algorithm of Reingold [274]). In particular, we perform the following transformations on the original graph:

Step 1: Regularization. We first transform the original graph G into an $O(1)$-regular graph G_1 such that (i) there is a one to one correspondence between connected components of G_1 and G, and (ii) mixing time of every connected component of G_1 is still $\text{polylog}(n)$ (using the fact that $\lambda = 1/\text{polylog}(n)$ and Proposition 6.3.1).

Step 2: Randomization. Next, we transform every connected component of G_1 to a random graph chosen from distribution of random graphs $\mathbb{G}$ to obtain a graph G_2. This transformation (w.h.p) preserves all connected components of G_1 and never merges two separate components of G_1 into G_2. As it turns out, the structure of random graphs (*beyond* their improved connectivity) makes them "easy to solve" for MPC algorithms (more on this below).

Step 3: Connectivity in random graphs. Finally, we design a novel algorithm for finding connected components of G_2 which are each a random graph sampled from $\mathbb{G}$. This algorithm can be seen as yet another transformation that reduces the diameter of each component to $O(1)$ and then solve the problem on a low-diameter graph using a simple broadcasting strategy.

We now elaborate more on each step of this algorithm.

Step 1: Regularization. The main steps of our algorithm heavily rely on the properties of *regular* graphs, for several important reasons that will become evident shortly. Our first goal is then to "regularize" the input while preserving its connected components, its spectral gap and the number of edges (Lemma 6.5.1). The standard procedure for regularizing a graph by adding self-loops to vertices (e.g., [274]) is too lossy for our purpose as it can dramatically reduce the spectral gap[4].

We instead use an approach based on the so-called *replacement product* (see, e.g., [276]): The idea is to replace each vertex v of the original graph with degree d_v, by a Δ-regular expander on d_v "copies of v", and then connect these expanders across according to edges of G to construct a $(\Delta+1)$-regular graph (see Section 6.5 for details). It is known that this

[4]Unlike vertex and edge expansion, spectral expansion is *not* a monotone property of edges of the graph.

product (approximately) preserves the spectral gap in the new graph, hence the mixing time of each component remains polylog(n) even after this transformation. Implementing this approach in the MPC model has its unique challenges as the degree of some vertices in the original graph can be as large as $\Omega(n)$ hence we need a *parallel* procedure for constructing the expanders and performing the product, as no machine can do these tasks locally on its $n^{\Omega(1)}$-size memory (see Lemmas 6.5.4 and 6.5.5).

We point out that replacement products have been used extensively in the context of connectivity and expansion to reduce the degree of *regular* graphs [276, 274], but to best of our knowledge, our (distributed) implementation of this technique for the regularization purpose itself, while preserving its spectral gap, is nontrivial (yet admittedly quite anticipated[5]). We believe this parallel regularization primitive itself will be a useful building block in future MPC graph algorithms.

Step 2: Randomization. The goal of the second step is, morally speaking, to replace each connected component of the regular graph G_1 with a purely *random graph* sampled from distribution $\mathbb{G}$ with degree $O(\log n)$ on the same connected component (which will indeed be connected with high probability by Proposition 6.3.3); This is the content of Lemma 6.6.1.

In order to achieve the desired transformation, we need to connect every vertex v in G_1 to $O(\log n)$ *uniformly random* vertices *in the same connected component as* v. The obvious challenge in this step is that the information about which vertices belong to the same connected component is decentralized and each machine only has a "local" view of the graph. To this end, we perform $O(\log n)$ *lazy random walks* of length $T = \text{polylog}(n)$ from *every vertex* of the graph G, where T is an upper bound on the mixing time of every connected component of G_1. This, together with the fact that G_1 is *regular*, ensures that the target of each random walk is (essentially) a uniformly random vertex in the corresponding connected component of G_1.

The main contribution in this step is an efficient parallel construction of a distributed data-structure for performing and manipulating *independent* random walks of length T in a regular graph, with only $O(\log T)$ MPC rounds; see Theorem 6.4. This allows us to perform the above transformation in $O(\log T) = O(\log \log n)$ MPC rounds. Standard ideas such as recursively computing random walks of certain length from every vertex in parallel and then "stitching" these walks together to double the length of each walk can be used to implement this step (see [15, 212, 173] for similar implementations in the PRAM model)[6].

[5] This in fact requires us to extend the proof of expansion of replacement product to non-regular graphs as all existing proofs of this result that we are aware of are assuming original graph is regular [276, 275, 278, 180, 291], while our sole purpose is to regularize the graph; see Section 6.5 for details.

[6] This step is also similar-in-spirit to streaming and distributed implementations of random walks in [283, 284], with the difference that we leverage the all-to-all communication of MPC model to achieve an *exponential* speed up of $O(\log T)$ rounds as opposed to $O(\sqrt{T})$ achieved by these works, which is tight [255].

The main challenge however, which is crucial for sampling from distribution $\mathbb{G}$, is that in all the aforementioned implementations, the random walks produced across vertices are *not independent of each other* as different walks become correlated once they "hit" the same vertex (the remainder of the walk would become the same for both walks).

A key observation that allows us to circumvent this difficulty is that in a *regular* graph, no vertex can become a "hub" which many different random walks hit (contrast this with a star-graph where every random walk almost surely hits the center); this is one of the key reasons that we need to perform the regularization step first. As such, many of the walks computed in the above procedure are indeed independent of each other. We use this observation along with several additional ideas (e.g., having each vertex compute multiple random walks and assign them randomly to different length walks in the recursive procedure above) to implement this step.

Step 3: Connectivity in random graphs. The final and main step of the proof is an algorithm for identifying all connected components of a graph which are each sampled from $\mathbb{G}$ in only $O(\log \log n)$ MPC rounds (Lemma 6.7.1). The centerpiece of this step is a leader-election based algorithm for connectivity (similar to most algorithms for sparse connectivity in the MPC model, e.g., [213, 218, 271]). A typical leader-election algorithm for connectivity would pick some set of "leader vertices" in each round, and let other non-leader vertices connect to some leader in their neighborhood. It then "contracts" every leader vertex and all non-leader vertices that choose this leader to connect, to form a component of the input graph. This way, components of the graph "grow" in each round as information propagates through leaders, until all components of the graph are discovered. The *rate of growth* of components in these algorithms is however typically only a *constant* as in general, it is hard to find components of size beyond a constant in each round (consider for instance the case when the underlying graph is a cycle). Consequently, $\Omega(\log n)$ rounds are necessary to find all connected components using these algorithms.

Our algorithm achieves an exponential speedup in rounds by crucially using the properties of the random graphs $\mathbb{G}$ to contract components which are *quadratically* larger after each round, i.e., it grows a component of size x into a component of size x^2 in each round.

The intuition behind the algorithm is as follows. Let $H \sim \mathbb{G}(n, d)$. Since H is essentially d-regular (Proposition 6.3.2), sampling each vertex as a leader with probability $\Theta(1/d)$, we expect each non-leader vertex to have a *constant* number of leader neighbors, say exactly 1 for simplicity. Since every vertex has d neighbors, contracting every leader vertex along with all of its non-leader neighbors into a single "mega-vertex" will form components of size d with total degree (roughly) d^2 in the contracted graph (this follows from the randomness in the distribution $\mathbb{G}$ as no single mega-vertex is likely to be the endpoint of more than one

of these d^2 edges). As such, the resulting graph after contraction is an almost d^2-regular random graph on n/d vertices. By continuing this process on the new graph, we can now pick each leader with probability $1/d^2$ instead and contract components of size d^2 (instead of d). Repeating this process i steps creates components of size d^{2^i} which implies that after $O(\log \log n)$ iterations we would be done. We stress that this algorithm exploits the "entropy" of the distribution $\mathbb{G}$ crucially, and not just the connectivity properties, e.g., expansion, of $\mathbb{G}$, hence it is not clear (and unlikely) that this algorithm can be made to work directly on expander graphs (i.e., without Step 2).

The outline above oversimplifies many details. Let us briefly mention two. Contracting vertices in this process *correlates* the edges of $\mathbb{G}$, impeding a recursive application. We bypass this problem by partitioning the edges of the random graph into $O(\log \log n)$ different batches and running the algorithm (and analysis) in each round of the computation using a "fresh random seed" (batch). This breaks the dependency between the choices made by the algorithm in previous rounds, and the randomness of the underlying graph. Another subtle issue is in the fact that the graphs in this process start "drifting" from regular to almost-regular with larger and larger discrepancy factors, indeed *exponentially larger* after each round. At some point, this discrepancy factor becomes so large that one cannot anymore continue the previous argument. Fortunately however, as we are only performing $O(\log \log n)$ rounds of computation, this only happens when size of each component has become $n^{\Omega(1)}$. At this point, we can simply stop the algorithm and argue that the diameter of the contracted graph is only $O(1)$. This allows us to run a simple algorithm for a BFS tree in this graph in $O(1)$ rounds, by computing levels of the tree one round at a time

6.5. Step 1: Regularization

We now show how to "preprocess" our graph in order to prepare it for the main steps of our algorithm, by turning it into a regular-graph without increasing its mixing time by much.

Lemma 6.5.1. *There exists an MPC algorithm that given any graph $G(V, E)$ computes another graph H with the following properties with high probability:*

1. *$|V(H)| = 2m$ and H is Δ-regular for some absolute constant $\Delta = O(1)$.*
2. *There is a one-to-one correspondence between the connected components of G and H.*
3. *Let H_i be a connected component of H corresponding to the connected component G_i of graph G. For any $\gamma < 1$, $T_\gamma(H_i) = O\Big(\frac{\log(n/\gamma)}{\lambda_2(G_i)}\Big)$.*

For any $\delta > 0$, the algorithm can be implemented on $O(m^{1-\delta})$ machines each with $O(m^\delta)$ memory and in $O(\frac{1}{\delta})$ MPC rounds.

To prove Lemma 6.5.1, we use an approach based on the standard replacement product described in the next section.

Replacement Product

Let G be a graph on n vertices $v_1, \ldots, v_n$ with degree d_v for $v \in V(G)$, and $\mathcal{H}$ be a family of n d-regular graphs $H_1, \ldots, H_n$ where H_v is supported on d_v vertices (we assume $d_v \geq d$ for all $v \in V(G)$). We construct the *replacement* product $G \textcircled{r} \mathcal{H}$ as follows:

- Replace the vertex v of G with a copy of H_v (henceforth referred to as a *cloud*). For any $i \in H_v$, we use (v, i) to refer to the i-th vertex of the cloud H_v.
- Let (u, v) be such that the i-th neighbor of u is the j-th neighbor of v. Then there exists an edge between vertices (u, i) and (v, j) in $G \textcircled{r} \mathcal{H}$. Additionally, for any $v \in V(G)$, if there exists an edge $(i, j) \in H_v$, then there exists an edge $((v, i), (v, j))$ in $G \textcircled{r} \mathcal{H}$.

It is easy to see that the replacement product $G \textcircled{r} \mathcal{H}$ is a $(d+1)$-regular graph on $2m$ vertices where m is the number of edges in G. The following proposition asserts that the spectral gap is preserved under replacement product.

Proposition 6.5.2 (cf. [276, 275])**.** *Suppose $\lambda_2(G) \geq \lambda_G$ and all $H_v \in \mathcal{H}$ are d-regular with $\lambda_2(H_v) \geq \lambda_H$. Then, $\lambda_2(G \textcircled{r} \mathcal{H}) = \Omega\left(d^{-1} \cdot \lambda_G \cdot \lambda_H^2\right)$*

Proposition 6.5.2 was first proved in [276] when G is also a D-regular graph and all copies in $\mathcal{H}$ are the same d-regular graph on D vertices (in fact, all proofs of this proposition that we are aware of, e.g., [276, 275, 278, 180, 291], are for this case). However, for our application, we crucially need this proposition for non-regular graphs G (after all, our ultimate goal is to "regularize" the graph). Nevertheless, extending these proofs to the case of non-regular graph G as stated in Proposition 6.5.2 is not hard albeit being rather technical and out of the scope of current thesis (see our paper [36] for this proof).

For our purpose, we only need Proposition 6.5.2 when every graph in $\mathcal{H}$ is a constant-degree regular expander. In this case, since $\lambda_H = \Omega(1)$ and $d = O(1)$, we obtain that the resulting graph $G \textcircled{r} \mathcal{H}$ has spectral gap at least $\lambda_2(G \textcircled{r} \mathcal{H}) = \Omega(\lambda_2(G))$.

Parallel Expander Construction

To use Proposition 6.5.2, we need to be able to create a family of expanders $\mathcal{H}$ in parallel over the set of vertices of the original graph G. This is a non-trivial task as degree of some vertices in G can be as high as $\Omega(n)$ and hence we need to create an expander with $\Omega(n)$ edges to replace them; at the same time, no single machine has $\Omega(n)$ memory to fit this expander and hence it should be constructed in parallel and distributed across multiple machines. We note that however, we can use a randomized algorithm for this task (i.e., we do not need necessarily an "explicit" construction).

Consider the following construction of a random d-regular undirected graph on n vertices for positive even integer d (allowing self-loops and parallel edges): Let $\pi_1, \ldots, \pi_{d/2}$ be $d/2$ permutations on $[n]$ which are independently and uniformly sampled from the set of all permutations. The resulting graph is H with $V(H) := [n]$ and

$$E(H) := \{(i, \pi_j(i)) : i \in [n], j \in [d/2]\} \tag{6.1}$$

for unordered pairs $(i, \pi_j(i))$. Let $\mathcal{G}_{n,d}$ be the probability space of the d-regular n-vertex graphs constructed in this way.

Proposition 6.5.3 (c.f. [149])**.** *Given a postive constant $\delta > 0$ and an positive even integer d, there is a constant c such that $\Pr_{H \sim \mathcal{G}_{n,d}}\left[\lambda_2(H) \geq 1 - \frac{2\sqrt{d-1}+\delta}{d}\right] \geq 1 - \frac{c}{n^\tau}$, where $\tau = \lceil(\sqrt{d-1}+1)/2\rceil - 1$.*

We choose d to be 100. By Proposition 6.5.3, we have

Corollary 6.3. *Let $d = 100$. There is a constant c such that for any positive integer n*

$$\Pr_{H \sim \mathcal{G}_{n,d}}\left[\lambda_2(H) \geq \frac{4}{5}\right] \geq 1 - \frac{c}{n^5}.$$

In the remaining of this section, we present an MPC algorithm to construct random d-degree graphs for a given sequence of positive integers $n_1, n_2, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i \leq 2m$.

RegularGraphConstruction$(m^\delta, n_1, \ldots, n_k)$. An algorithm for constructing random d-regular graphs with $n_1, n_2, \ldots, n_k$ vertices for $d = 100$ on machines of memory $O(m^\delta)$.

Output: d-regular graphs $\overline{H}_{n_i}$ for $1 \leq i \leq k$.

1. For every $n_i \leq m^\delta$ **in parallel** repeat the following process until the resulting graph $\overline{H}_{n_i}$ satisfies $\lambda_2(\overline{H}_{n_i}) \geq 4/5$: uniformly sample $d/2$ permutations $\pi_1, p_2, \ldots \pi_{d/2}$ on $[n_i]$, and construct graph $\overline{H}_{n_i}$ by Eq. (6.1).
2. For every $n_i > m^\delta$ **in parallel** construct $\overline{H}_{n_i}$ on $d \cdot \lceil n_i/m^\delta \rceil$ machines
   (a) Independently and uniformly sample $v_{n_i,j,k}$ from $[n^{10}]$ for all $j \in [n_i], k \in [d/2]$.
   (b) For every $k \in [d/2]$, sort $\{v_{n_i,1,k}, \ldots, v_{n_i,n_i,k}\}$, and set $\pi_{n_i,k}(j)$ to be α if $v_{j,k}$ is α-th largest number among $\{v_{n_i,1,k}, \ldots, v_{n_i,n_i,k}\}$.
   (c) Construct graph $\overline{H}_{n_i}$ using $\pi_{n_i,1}, \ldots, \pi_{n_i,d/2}$ with edge set specified in Eq. (6.1).

We use RegularGraphConstruction to prove the following lemma.

Lemma 6.5.4. *There exists an MPC algorithm that given a sequence of positive integers $n_1, n_2, \ldots, n_k$ satisfying $\sum_{i=1}^{k} n_i \leq 2m$, with high probability, computes a set of graphs $\overline{H}_{n_1}, \overline{H}_{n_2}, \ldots, \overline{H}_{n_k}$ such that for every $\overline{H}_{n_i}$ for $1 \leq i \leq k$, $\overline{H}_{n_i}$ is a d-degree regular graph with $\lambda_2(\overline{H}_{n_i}) \geq 4/5$.*

For any $\delta > 0$, the algorithm can be implemented with $O(m^{1-\delta})$ machines each with $O(m^{\delta})$ memory, and in $O(1/\delta)$ MPC rounds.

Proof. We first show the correctness of the RegularGraphConstruction algorithm. By Proposition 6.5.3, step 1 construct regular graphs with desirable spectral gap with high probability for every $n_i \leq m^{\delta}$. Now we show that step 2 construct regular graphs with desirable spectral gap with high probability for every $n_i > m^{\delta}$ as well. For every $n_i \geq m^{\delta}$ and $k \in [d/2]$, the probablity that $v_{n_i,1,k}, \ldots, v_{n_i,n_i,k}$ are distinct is at least

$$1 \cdot 2 \cdot \ldots \left(1 - \frac{n_i - 1}{n^{10}}\right) > \left(1 - \frac{n_i}{n^{10}}\right)^{n_i} \geq 1 - \frac{n_i^2}{n^{10}} \geq 1 - \frac{1}{n^8}.$$

If $v_{n_i,1,k}, \ldots, v_{n_i,n_i,k}$ are distinct, then $\pi_{n_i,k}$ is a random permutation among all the permutations on $[n_i]$, since all the permutations are constructed with same probability. Conditioned on this, $\overline{H}_{n_i}$ is a graph sampled from $\mathcal{G}_{n,d}$. By union bound, with probability $1 - \frac{1}{n^7}$, step 2(c) obtain $\overline{H}_{n_i} \sim \mathcal{G}_{n_i,d}$ for every $n_i > m^{\delta}$. By Corollary 6.3, if $\overline{H}_{n_i} \sim \mathcal{G}_{n_i,d}$, then $\lambda_2(\overline{H}_{n_i}) \geq 4/5$ with probability $1 - \frac{c}{n_i^5}$ for some constant $c \geq 0$. By union bound, all the $\overline{H}_{n_i}$ constructed satisfying $\lambda_2(\overline{H}_{n_i}) \geq 4/5$ with probability at least

$$1 - \sum_{\ell=n^{\varepsilon}}^{n} \frac{c}{\ell^5} \geq 1 - O\left(\frac{\log n}{n^4}\right).$$

Hence, the algorithm gives us graphs with desirable spectral gap with probability at least $1 - \frac{1}{n^3}$.

In the implementation of step 1, we assign every $n_i \leq m^{\delta}$ to a single machine such that for every machine, the sum of n_i assigned to it is at most $O(m^{\delta})$. Hence, step 1 can be done in $O(1)$ MPC rounds.

In step 2, for each n_i and $k \in [d/2]$, we use $\lceil n_i/m^{\delta} \rceil$ machines to sample $v_{n_i,j,k}$ for all the $j \in [n_i]$ in $O(1)$ MPC rounds. Sorting $\{v_{n_i,1,k}, \ldots, v_{n_i,n_i,k}\}$ can be done in $O(1/\delta)$ MPC rounds on the same machines (see Section 1.1.3). Then $\pi_{n_i,k}(j)$ and thus edges of $\overline{H}_{n_i}$ can be computed locally after sort. This concludes the proof. □

Parallel Replacement Product

We present an MPC implementation of replacement product $G \textcircled{r} \mathcal{H}$, where $\mathcal{H} = \{H_v : v \in V\}$ is defined as follows: For every $v \in V$, H_v is a copy of $\overline{H}_{d_v}$, where $\overline{H}_{d_v}$ are the d-degree regular graphs with d_v vertices returned by $\mathsf{RegularGraphConstruction}(m^\delta, d_{v_1}, \ldots, d_{v_n})$.

Lemma 6.5.5. *Given a graph $G(V, E)$ and $\overline{H}_{d_v}$ for every $v \in V$, there is an MPC algorithm to compute $G \textcircled{r} \mathcal{H}$, where $\mathcal{H} = \{H_v : v \in V\}$ such that H_v is a copy of $\overline{H}_{d_v}$ for every $v \in V$.*

The algorithm can be implemented with $O(m^{1-\delta})$ machines each with $O(m^\delta)$ memory, and in $O(1/\delta)$ MPC rounds.

Lemma 6.5.5 is obtained by the definition of replacement product and the following algorithm.

$\mathsf{ReplacementProduct}(G, \{\overline{H}_{d_v} \text{ for every } v \in V(G)\})$. An algorithm for constructing $G \textcircled{r} \mathcal{H}$.

Output: $H := G \textcircled{r} \mathcal{H}$.

1. For every $v \in V(G)$ **in parallel** set H_v be a copy of $\overline{H}_{d_v}$, and let H be initially $\cup_{v \in V(G)} H_v$.
2. For every edge $(u, v) \in E$ **in parallel** where v is i-th neighbor of u and u is j-th neighbor of v in G, add an edge to H between i-th vertex of H_u and j-th vertex of H_v.
3. Return H

The proof of correctness of this algorithm is straightforward.

Proof of Lemma 6.5.1

Proof. By Lemma 6.5.4 and Lemma 6.5.5, we can compute the replacement product $H := G \textcircled{r} \mathcal{H}$ where $\mathcal{H}$ is a family of graphs such that for all $v \in V(G)$, $\lambda_2(H_v) \geq 4/5$. By definition of replacement product and since $d = O(1)$, we obtain that $|V(H)| = 2m$ and H is Δ-regular for $\Delta = d + 1 = O(1)$. This proves the first part of the lemma.

Consider any connected component G_i of G and define $\mathcal{H}_i := \{H_v \in \mathcal{H} : v \in V(G_i)\}$. It is immediate to see that the subgraph of H induced on vertices of $V(G_i) \times V(H_v)$ for $v \in V(G_i)$ (informally speaking, the vertices added to H because of G_i) is exactly $G_i \textcircled{r} \mathcal{H}_i$ which we denote by H_i. As replacement product preserves connectivity, H_i is a connected component of H, hence proving the second part of the lemma.

Finally, as $H_i = G_i \textcircled{r} \mathcal{H}_i$ and $\lambda_2(H_v) = \Omega(1)$ for all $H_v \in \mathcal{H}_i$, by Proposition 6.5.2,

$\lambda_2(H_i) = \Omega(\lambda_2(G_i))$ (recall that $d = O(1)$). As such, by Proposition 6.3.1, mixing time

$$T_\gamma(H_i) = O\Big(\frac{\log(n/\gamma)}{\lambda_2(H_i)}\Big) = O\Big(\frac{\log(n/\gamma)}{\lambda_2(G_i)}\Big),$$

for any $\gamma < 1$, concluding the proof of the third part. Implementation details of the algorithm follow from Lemmas 6.5.4 and 6.5.5. □ Lemma 6.5.1

6.6. Step 2: Randomization

We present the second step of our algorithm in this section. Roughly speaking, this step transforms each connected component of the graph into a "random graph" (according to the definition of distribution $\mathbb{G}$ in Section 6.3) on the same set of vertices. Formally,

Lemma 6.6.1. *Suppose $G(V, E)$ is any n-vertex Δ-regular graph such that $T_{\gamma^*}(G_i) \leq T$ for $\gamma^* := n^{-10}$ and for all connected component G_i of G. There exists an MPC algorithm that given G and integer T computes another graph H with the following properties with high probability:*

1. *$V(H) = V(G)$, $|E(H)| = O(n)$ and each connected component G_i of G corresponds to a connected component H_i of H on $V(H_i) = V(G_i)$.*
2. *The connected component H_i of H is a random graph on $n_i = |V(H_i)|$ vertices sampled from the distribution $\textsc{dist}(H_i)$ such that $|\textsc{dist}(H_i) - \mathbb{G}(n_i, 100\log n)|_{tvd} = n^{-8}$.*

For any $\delta > 0$, the algorithm can be implemented with $O(T^2 \cdot n^{1-\delta} \cdot \Delta \log^2 n)$ machines each with $O(n^\delta)$ memory and in $O(\frac{1}{\delta} \cdot \log T)$ MPC rounds.

We point out that the choice of constant 100 in $\mathbb{G}(n_i, 100 \cdot \log n)$ in Lemma 6.6.1 is arbitrary and any sufficiently large constant (say larger than 8) suffices for our purpose (similarly also for the exponent of n^{-1} in γ^*).

To prove Lemma 6.6.1, we design a general algorithm for performing *independent* random walks in the MPC model which can be of independent of interest. Let $G(V, E)$ be a Δ-regular graph and $W = \Delta^{-1} \cdot A$ be its random walk matrix (note that this is scalar product with Δ^{-1} as G is Δ-regular). For any vertex $u \in V$, and integer $t \geq 1$, the vector $W^t \cdot \boldsymbol{e}_u$ denotes the distribution of a random walk of length t starting from u where $\boldsymbol{e}_u$ is an n-dimensional vector which is all zero except for the entry u which is one. We use $\mathcal{D}_{\mathsf{RW}}(u, t) = W^t \cdot \boldsymbol{e}_u$ to denote this distribution.

Theorem 6.4. *There exists an MPC algorithm that given any Δ-regular graph $G(V, E)$ and integer $t \geq 1$, outputs a vector $(v_1, \ldots, v_n)$ such that with high probability:*

1. *For any $i \in [n]$, v_i is sampled from $\mathcal{D}_{\mathsf{RW}}(u_i, t)$, where u_i is the i-th vertex in V.*
2. *The choice of v_i is* independent *of all other vertices v_j. In other words, $(v_1, \ldots, v_n)$ is sampled from the product distribution $\bigotimes_{i=1}^{n} \mathcal{D}_{\mathsf{RW}}(u_i, t)$*

For any $\delta > 0$, the algorithm can be implemented with $O(t^2 \cdot n^{1-\delta} \cdot \Delta \log n)$ machines each with $O(n^{\delta})$ memory and in $O(\frac{1}{\delta} \cdot \log t)$ MPC rounds.

6.6.1. Proof of Theorem 6.4: The Random Walk Algorithm

We start by presenting a parallel algorithm for proving Theorem 6.4 without getting into the exact details of its implementation, and then present an MPC implementation of this parallel algorithm. We start by introducing a key data structure in our algorithm.

Layered Graph

A key component of our algorithm in Theorem 6.4 is the notion of a *layered graph* which we define in this section and present its main properties.

Definition 6.1 (Layered Graph)**.** *For a graph $G(V, E)$ and integer $t \geq 1$, the layered graph $\mathcal{G}(G, t)$ of G is defined as the following* directed *graph:*

1. ***Vertex-set:*** *The vertex-set $\mathcal{V}$ of $\mathcal{G}$ is the set of all triples $(u, i, j) \in V \times [2t] \times [t+1]$.*
2. ***Edge-set:*** *There is a directed edge $(u, i, j) \to (v, \ell, k)$ in $\mathcal{G}$ whenever $(u, v) \in E$ and $k = j + 1$ for all choice of i and ℓ.*

Throughout the chapter, we use greek letters to denote the vertices in the layered graph.

For any vertex $\alpha = (u, i, j) \in \mathcal{V}$, we define $v(\alpha) = u \in V$. We partition the set of vertices $\mathcal{V}$ into $t+1$ sets $\mathcal{V}_1, \ldots, \mathcal{V}_{t+1}$ where the j-th set consists of all vertices (u, i, j) for $u \in V$ and $i \in [2t]$. We refer to each set $\mathcal{V}_j$ as a *layer* of the graph $\mathcal{G}$. It is immediate to see that $\mathcal{G}$ consists of $t+1$ layers and all edges in $\mathcal{G}$ are going from one layer to the next. Additionally, any vertex $u \in V$, contains $2t$ "copies" in every layer. As such, any edge in E is mapped to t directed bi-cliques on the $2 \cdot 2t$ copies of its endpoints between every two consecutive layers of $\mathcal{G}$.

Paths and walks in $\mathcal{G}$ and G: The main property of the layered graph $\mathcal{G}$ that we use is that any path starting from $\mathcal{V}_1$ and ending in $\mathcal{V}_{t+1}$ in $\mathcal{G}$ corresponds to a walk of length t (but not necessarily a path) in G. More formally, consider a path $\mathcal{P}_\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{t+1}$ where $\alpha = \alpha_1$ belongs to $\mathcal{V}_1$. We can associate to $\mathcal{P}_\alpha$ a walk of length t in G starting from the vertex $v = v(\alpha)$, denoted by $W(\mathcal{P}_\alpha)$, in a straightforward way by traversing the vertices $u_i = v(\alpha_i)$ for $\alpha_i \in \mathcal{P}_\alpha$.

Sampled layered graph. In our algorithm, we work with a random subgraph of the

layered graph defined as follows: For any vertex in $\mathcal{G}$ *independently*, we pick exactly one of its *outgoing* edges *uniformly at random* to form a subgraph $\mathcal{G}_{\mathsf{S}}$, referred to as the *sampled layered graph*.

As the out-degree of any vertex in $\mathcal{G}_{\mathsf{S}}$ is exactly one, starting from any vertex $\alpha \in \mathcal{V}_1$, there is a *unique* path $\mathcal{P}_\alpha$ of length t in $\mathcal{G}_{\mathsf{S}}$ from α to some vertex $\beta \in \mathcal{V}_{t+1}$. It is easy see that a path $\mathcal{P}_\alpha$ in $\mathcal{G}_{\mathsf{S}}$ corresponds to a *random* walk of length t in $\mathcal{G}$ starting from the vertex $v(\alpha)$ and ending in $v(\beta)$ (the randomness comes from the choice of $\mathcal{G}_{\mathsf{S}}$). We have the following key observation.

Observation 6.6.2. *Suppose $\mathcal{P}_{\alpha_1}, \ldots, \mathcal{P}_{a_k}$ are k* vertex disjoint paths *from $\mathcal{V}_1$ to $\mathcal{V}_{t+1}$ in $\mathcal{G}_{\mathsf{S}}$. Then, the associated walks $W(\mathcal{P}_{\alpha_1}), \ldots, W(\mathcal{P}_{\alpha_k})$ form k* independent *random walks of length t in G.*

The justification for Observation 6.6.2 is the simple fact that vertex disjoint paths in $\mathcal{G}_{\mathsf{S}}$ do not share any randomness in choice of their neighbors.

In the rest of this section, we show that a sampled layered graph contains $\Omega(n)$ vertex disjoint paths from the first layer to the last one with high probability. Intuitively, this allows us to "extract" $\Omega(n)$ independent random walks from a sampled layered graph. We then use this fact in the next section to design our algorithm for simulating independent random walks in G.

Define $\mathcal{V}_1^* \subseteq \mathcal{V}_1$ as the set of all vertices $(v, 1, 1) \in \mathcal{V}_1$ for $v \in V$. We prove that the $\Omega(n)$ vertex disjoint paths mentioned above can all be starting from vertices in $\mathcal{V}_1^*$. Formally,

Lemma 6.6.3. *For any vertex $\alpha \in \mathcal{V}_1^*$, $\mathcal{P}_\alpha$ in $\mathcal{G}_{\mathsf{S}}$ is vertex disjoint from $\mathcal{P}_\beta$ for all $\beta \neq \alpha \in \mathcal{V}_1^*$, with probability at least $1/2$.*

We emphasize that in Lemma 6.6.3, the paths starting from $\mathcal{V}_1^*$ are *only* guaranteed to be vertex disjoint with constant probability from other paths starting from $\mathcal{V}_1^*$ and not all of $\mathcal{V}_1$. Before getting into the proof of Lemma 6.6.3, we prove the following auxiliary claim regarding the number of paths of certain lengths in $\mathcal{G}$ (not in $\mathcal{G}_{\mathsf{S}}$).

Claim 6.6.4. *For any layer $j \in [t+1]$ and any vertex $\alpha \in \mathcal{V}_j$, the number of paths in $\mathcal{G}$ that start from some vertex in $\mathcal{V}_1^*$ and end in vertex α is $P_j = \left(\Delta^{j-1} \cdot (2t)^{j-2}\right)$.*

Proof. Let $\alpha = (v, i, j)$ be in layer j. Since G is Δ-regular, $v \in V$ has exactly Δ neighbors in V. By construction of $\mathcal{G}$, this means that v has $\Delta \cdot (2t)$ neighbors in $\mathcal{V}_{j-1}$ and hence there are $\Delta \cdot (2t)$ paths of length 1 that end up in α. Similarly, the starting point of any of these paths has exactly $\Delta \cdot (2t)$ neighbors in $\mathcal{V}_{j-2}$ and hence there are $(\Delta \cdot 2t)^2$ paths of length 2 that can end up in α. Continuing this inductively, we obtain that there are $(\Delta \cdot (2t))^{j-1}$

paths of length j that can reach the vertex α. By the layered structure of the graph $\mathcal{G}$, it is clear that all these paths need to start from a vertex in $\mathcal{V}_1$.

Furthermore, if $(u, i, 1)$ (for some $u \in V$ and $i \in [2t]$) is starting point one of these paths, then for all $\ell \in [2t]$, $(u, \ell, 1)$ would also be a starting point of one such path (this is because neighborset of all vertices $(u, \ell, 1)$ is the same). As such, exactly $1/(2t)$ fraction of these starting points belong to $\mathcal{V}_1^*$ and hence there are $P_j := \left(\Delta^{j-1} \cdot (2t)^{j-2}\right)$ paths in $\mathcal{G}$ that start from a vertex in $\mathcal{V}_1^*$ and end in vertex α. □ Claim 6.6.4

Proof of Lemma 6.6.3. Let $\mathcal{P}_\alpha = \alpha_1, \alpha_2, \ldots, \alpha_{t+1}$ where $\alpha_1 = \alpha$ and each α_j belongs to $\mathcal{V}_j$ for $j > 1$. We define the following $t+1$ random variables $X_1, \ldots, X_{t+1}$, where X_j counts the number of paths that start from a vertex $\beta \neq \alpha \in \mathcal{V}_1^*$ and contain vertex α_j (as their j-th vertex). In other words, X_j counts the number of paths that "hit" $\mathcal{P}_\alpha$ in layer $\mathcal{V}_j$.

Clearly, $X_1 = 0$. For any $j > 1$, we define indicator random variables $Y_{j,1}, Y_{j,2}, \ldots, Y_{j,P_j}$ where P_j (the quantity bounded in Claim 6.6.4) is the number of paths that start from $\mathcal{V}_1^*$ and end in α_j in $\mathcal{G}$: for all $i \in [P_j]$, $Y_{j,i} = 1$ iff the i-th path (according to any arbitrary ordering) is fully appearing in $\mathcal{G}_{\mathsf{S}}$ as well. Clearly, $X_j = \sum_i Y_{j,i}$. Hence, by linearity of expectation,

$$\mathbb{E}[X_j] = \sum_{i=1}^{P_j} \Pr(Y_{j,i} = 1) = |P_j| \cdot \left(\frac{1}{\Delta \cdot (2t)}\right)^{j-1} \underset{\text{Claim 6.6.4}}{=} \frac{1}{2t}. \tag{6.2}$$

The second equality above is because in $\mathcal{G}_{\mathsf{S}}$, each edge in the path has probability of $\frac{1}{\Delta \cdot (2t)}$ to appear (as out-degree of any vertex in $\mathcal{G}$ is $\Delta \cdot (2t)$ and we are picking one of these edges uniformly at random in $\mathcal{G}_{\mathsf{S}}$; moreover, the edges of a path appear independently in $\mathcal{G}_{\mathsf{S}}$).

Finally, notice that $X := \sum_{j=1}^{t+1} X_j$ counts the total number of paths starting from vertices in $\mathcal{V}_1^*$ that can ever "hit" $\mathcal{P}_\alpha$ in any layer. Hence, $\mathbb{E}[X] = 1/2$ by Eq (6.2) (recall that $X_1 = 0$) and by Markov bound, $\Pr(X = 1) \leq 1/2$. This implies that with probability at least $1/2$, $\mathcal{P}_\alpha$ is vertex disjoint from other paths starting from a vertex in $\mathcal{V}_1^*$. □ Lemma 6.6.3

A Parallel Random Walk Algorithm

We now present a parallel algorithm for performing independent random walks of fixed length from every vertex of the graph. We start by presenting an algorithm with a weaker guarantee: in this algorithm only $\Omega(n)$ vertices are able to achieve a *truly* independent random walk destination; moreover, these vertices are unknown to the algorithm. We then present a subroutine for detecting these $\Omega(n)$ vertices. Finally, we combine these two subroutines to obtain our final algorithm.

Recall that for any vertex $u \in V(G)$ and integer $t \geq 1$, $\mathcal{D}_{\mathsf{RW}}(u, t)$ is the distribution of

a random walk of length t from u. We present the following algorithm.

$\mathsf{SimpleRandomWalk}(G,t)$. An algorithm for performing a random walk of length t from every vertex in a given graph G.

Output: For any vertex $u_i \in V(G)$, a vertex $v_i \in V(G)$ such that $v_i \sim \mathcal{D}_{\mathsf{RW}}(u_i,t)$.

1. Randomly sample a sampled subgraph $\mathcal{G}_{\mathsf{S}}$ from the layered graph $\mathcal{G}(G,t)$.
   (a) Set $\mathcal{V} = V(G)\times[2t]\times[t+1]$, and distribute the vertices of $\mathcal{V}$ to all the machines such that each machine contains $O(n^{\delta})$ vertices.
   (b) For every vertex $\alpha = (v,i,j) \in \mathcal{V}$ such that $j \leq t$ **in parallel** independently and uniformly sample a number n_α from $[\Delta]$ and i_α from $[2t]$.
   (c) Set $\mathcal{G}_{\mathsf{S}}$ to be empty initially.
   (d) For every vertex $\alpha = (v,i,j) \in \mathcal{V}$ such that $j \leq t$ **in parallel** set v_α to be n_α-th neighbor of v in G, and add an edge from α to $(v_\alpha, i_\alpha, j+1)$ to $\mathcal{G}_{\mathsf{S}}$.
2. For any vertex $\alpha \in \mathcal{G}_{\mathsf{S}}$, define $N_0(\alpha) = \beta$ where $(\alpha,\beta) \in \mathcal{G}_{\mathsf{S}}$ is the (only) outgoing edge of α in $\mathcal{G}_{\mathsf{S}}$ (define $\beta = \perp$ if α belongs to $\mathcal{V}_{t+1}$ and hence has no outgoing edge).
3. For $i = 1$ to $\log t$ phases: For every $\alpha \in \mathcal{G}_{\mathsf{S}}$ **in parallel** let $N_i(\alpha) = N_{i-1}(N_{i-1}(\alpha))$ (assuming $N_{i-1}(\perp) = \perp$).
4. For any $\alpha \in \mathcal{V}_1^*$, return $v = v(N_{\log t}(\alpha))$ as the target of the vertex $u = v(\alpha)$ (recall that $\mathcal{V}_1^*$ is the set of all vertices $(u,1,1) \in \mathcal{V}$ for $u \in V(G)$).

We first have the following simple claim.

Claim 6.6.5. *For any vertex $\alpha \in \mathcal{V}_1^*$ of $\mathcal{G}_{\mathsf{S}}$, $N_{\log t}(\alpha)$ is the endpoint of the path $\mathcal{P}_\alpha$ in $\mathcal{G}_{\mathsf{S}}$.*

Proof. We prove by induction that $N_i(\alpha)$ is the vertex at distance 2^i from α in $\mathcal{P}_\alpha$. The base case for $i = 0$ is true as $N_0(\alpha) = \beta$ where β is the endpoint of the outgoing edge of α. For $i > 0$, by induction, $N_{i-1}(\alpha)$ is the vertex θ at distance 2^{i-1} from α and $N_{i-1}(\theta)$ is the vertex at distance 2^{i-1} from θ. Hence $N_i(\alpha) = N_{i-1}(N_{i-1}(\alpha))$ is at distance 2^i from α (as $\mathcal{G}_{\mathsf{S}}$ is a directed acyclic graph with edges going only from one layer to the next). As such, $N_{\log t}(\alpha)$ is at distance t from α and hence is the endpoint of the path $\mathcal{P}_\alpha$. □

We say that $\mathsf{SimpleRandomWalk}(G,t)$ *finds* the vertex v for u if v is returned as the target vertex of u. Claim 6.6.5 combined with Observation 6.6.2 already implies that for any vertex $u \in V(G)$, the vertex v found by SimpleRandomWalk is distributed according to $\mathcal{D}_{\mathsf{RW}}(u,t)$. We further have,

Lemma 6.6.6. *For any $u \in G$, $\mathsf{SimpleRandomWalk}(G,t)$ finds $v \sim \mathcal{D}_{\mathsf{RW}}(u,t)$ such that w.p. at least $1/2$, v is* independent *of all other vertices found by* SimpleRandomWalk.

Proof. Follows immediately from Claim 6.6.5, Observation 6.6.2, and Lemma 6.6.3. □

By Lemma 6.6.6, we are able to find $\Omega(n)$ independent random walks in G with high probability. However, a-priori it is not obvious how to detect these walks. In the following, we briefly describe a simple parallel procedure for this task.

Detecting independent random walks. The idea is to first find the path $\mathcal{P}_\alpha$ for every $\alpha \in \mathcal{V}_1^*$ and then remove any $v(\alpha)$ from consideration if $\mathcal{P}_\alpha$ intersects another path *starting from* $\mathcal{V}_1^*$. To do this, we need the following recursive "marking" procedure for marking all vertices on a path $\mathcal{P}_\alpha$:

$\mathsf{Mark}(\alpha, \beta, k)$: An algorithm for marking all vertices in the path $\mathcal{P}_\alpha$ recursively.

1. Mark the vertex $\beta \in \mathcal{V}$ with label α.
2. If $k = 0$ stop. Otherwise recurse on $\mathsf{Mark}(\alpha, \beta, k-1)$ and $\mathsf{Mark}(\alpha, N_{k-1}(\beta), k-1)$.

It is easy to see that by running $\mathsf{Mark}(\alpha, \alpha, \log t)$ for every $\alpha \in \mathcal{V}_1^*$ we can mark all vertices across all paths $\mathcal{P}_\alpha$ (this can be proven inductively using an argument similar to Claim 6.6.5). We remove any path $\mathcal{P}_\alpha$ which contains a vertex which is marked by more than one vertex. This way, all remaining paths are going to be vertex disjoint from each other and hence correspond to independent random walks.

We show that Mark algorithm can be implemented in parallel for all the vertices in $\mathcal{V}_1^*$, and is used to identify all the independent random walks.

DetectIndependence : An algorithm for detecting independent random walks for $\mathcal{V}_1^*$.

1. Set $S_{\log t} = \emptyset$ initially.
2. For every $\alpha \in \mathcal{V}_1^*$ **in parallel** add (α, α) to $S_{\log t}$.
3. For $k = \log t, \log t - 1, \ldots, 1$:
   (a) Set $S_{k-1} = \emptyset$ initially.
   (b) For every $(\alpha, \beta) \in S_k$ **in parallel** add (α, β) to S_{k-1}, and add $(\alpha, N_k(\beta))$ to S_{k-1} if $N_k(\beta) \neq \perp$.
4. Let T be the set of β such that there are $\alpha_1 \neq \alpha_2$ such that both (α_1, β) and (α_2, β) are in S_0 (by sorting all the pairs in S_0 according to the second coordinate).
5. Return the set $\{\alpha : \nexists \beta \text{ s.t. } (\alpha, \beta) \in S_0, \beta \in T\}$.

By the description of Algorithm DetectIndependence and since sorting can be done in $O(1/\delta)$ rounds if memory per machine is $O(n^\delta)$, we obtain the following claim.

Claim 6.6.7. *Algorithm* DetectIndependence *returns a set of vertices in* $\mathcal{V}_1^*$ *such that* α *is in the set iff* P_α *is an independent random walk for any* $\alpha \in \mathcal{V}_1^*$.

Algorithm DetectIndependence *requires* $O(t^2 \cdot n^{1-\delta})$ *machines each with* $O(n^\delta)$ *memory and* $O(\frac{1}{\delta} \cdot \log t)$ *MPC rounds.*

Proof of Theorem 6.4. We simply run SimpleRandomWalk(G, t) in parallel $\Theta(\log n)$ times and detect the independent random walks found by each run using the marking procedure above. By Lemma 6.6.6, with probability $1/2$ we are able to find an independent random walk for any fixed vertex in each of the $\Theta(\log n)$ trials. Hence, with high probability, we are able to find an independent random walk for every vertex of G. This concludes the proof of correctness of the algorithm.

We now briefly describe the MPC implementation details of this algorithm. To implement SimpleRandomWalk(G, t), we first create the vertex-set of the graph of $\mathcal{G}(G, t)$ which consists of $O(n \cdot t^2)$ vertices. We make every vertex responsible for maintaining the $O(\Delta \cdot t)$ of its neighbors and performing the random walks (the information needed by any single vertex resides entirely on one machine). Sampling $\mathcal{G}_{\mathsf{S}}$ is then straightforward. The rest of the algorithm can also be implemented in a straightforward way by spending $O(1/\delta)$ rounds for each iteration of for-loop in Line (3) of SimpleRandomWalk. By Claim 6.6.7, DetectIndependence also needs $O(\log t/\delta)$ rounds. Hence, in total, we only need $O(\log t/\delta)$ MPC rounds to implement the algorithms.

As for the memory per machine, for any fixed vertex, we only need $O(\Delta)$ (as opposed to $O(\Delta \cdot t)$) on the machine this vertex resides to sample an edge from $\mathcal{G}_{\mathsf{S}}$ as the $O(\Delta \cdot t)$ neighbors of any vertex in $\mathcal{G}(G, t)$ can be described by only $O(\Delta)$ edges (the rest are copies of the same edge to multiple copies of the same vertex on the next layer). We further need to store $O(\log t)$ intermediate vertices in $N(\cdot)$ and so each vertex needs $O(\Delta + \log t)$ memory and we have $O(n \cdot t^2 \cdot \log n)$ vertices in total (recall that we are performing $O(\log n)$ parallel random walks), finalizing the proof. □

6.6.2. Proof of Lemma 6.6.1: The Randomization Step

We now use Theorem 6.4 to prove Lemma 6.6.1. In Lemma 6.6.1, we need to perform lazy random walks, while Theorem 6.4 is performing random walks. However, this is quite easy to fix: we simply add Δ self-loops to every vertex of G. This makes the graph 2Δ regular while ensuring that the distribution of a random walk in the new graph corresponds to a lazy random walk in the original graph. We use $\widetilde{G}$ to refer to this new 2Δ-regular graph.

Proof of Lemma 6.6.1. We construct the graph $\widetilde{G}$ as specified and run algorithm in Theorem 6.4 on $\widetilde{G}$ for random walks of length T for $k = 50 \log n$ times in parallel. In the

following, we condition on the high probability event that the random walk algorithm succeeds. The graph H is defined as follows: $V(H) = V(\widetilde{G}) = V(G)$; for any $u \in V(H)$, connect u to the k vertices $v_{u,1}, \ldots, v_{u,k}$ found by the random walk algorithm for $u \in V(\widetilde{G})$. We now establish the desired properties of H.

Let G_i be any connected component of G. Any vertex $u \in V(G_i)$ in H is connected to k vertices in $V(G_i)$ in H: this is because a lazy random walk starting from a vertex in $V(G_i)$ cannot "escape" the component G_i. As such, any vertex $u \in V(G_i)$ is connected to k vertices in $V(G_i)$. Hence, the distribution of H_i is a graph in which every vertex is connected to $k = 50 \log n$ other vertices in $V(H_i)$ chosen according to the distribution of a lazy random walk of length T in graph G_i. The distribution $\mathbb{G}(n_i, 100 \log n)$ is a distribution on which every vertex in $V(H_i)$ is connected to $(100 \log n/2) = k$ vertices in $V(H_i)$ chosen uniformly at random. Since we are performing lazy random walks of length at least $T_{\gamma^*}(G_i)$, we expect these two distributions to be close to each other. Formally, let $\mathcal{U}_{V(H_i)}$ denote the uniform distribution on $V(H_i)$. We have,

$$|\textsc{dist}(H_i) - \mathbb{G}(n_i, 100 \log n)|_{tvd} \leq \sum_{u \in V(\widetilde{G}_i)} \left|\mathcal{D}_{\mathsf{RW}}(u, T) - \mathcal{U}_{V(H_i)}\right|_{tvd} \leq n_i \cdot 1/n^{10} \leq 1/n^9.$$

This proves the second part of the lemma. To prove the first part of the lemma we need to prove that each H_i is connected with high probability. This follows because H_i has a similar distribution as $\mathbb{G}(n_i, 100 \log n)$ and a graph sampled from $\mathbb{G}(n_i, 100 \log n)$ is connected with probability at least $1 - 1/n^{25}$ by Proposition 6.3.3 (by setting $d = 100 \log n \geq 100 \log n_i$ and assuming $n_i \geq 2$ as G contains no isolated vertices), and hence by Fact 2.6.7 H_i is also connected with probability at least $1 - 1/n^{25} - 1/n^9$, finalizing the proof of correctness.

The number of machines needed by this algorithm is $O(\log n)$ times the number of machines in Theorem 6.4 for $t = T$ and the memory per machine is the same. □

6.7. Step 3: Connectivity in Random Graphs

In this section we present the final and paramount step of our algorithm, which involves finding connected components of a collection of disjoint random graphs chosen from $\mathbb{G}$.

Lemma 6.7.1. *Let $G(V, E)$ be a graph on n vertices such that any connected component G_i of G with $n_i = |V(G_i)|$ is sampled from $\mathbb{G}(n_i, 100 \log n)$. There exists an MPC algorithm which identifies all connected components of G with high probability (over both the randomness of the algorithm and the distribution $\mathbb{G}$).*

For any $\delta > 0$, the algorithm can be implemented with $O(n^{1-\delta}) \cdot \text{polylog}(n)$ machines each with $O(n^{\delta}) \cdot \text{polylog}(n)$ memory and $O(\frac{1}{\delta} \cdot \log \log n)$ MPC rounds.

During the course of our exposition in this section, we need to set many parameters which we collect here for convenience.

$$\begin{aligned}
\varepsilon &:= (100 \cdot \log n)^{-2} && : \text{ used to bound the discrepancy factor of almost-regular graphs,} \\
s &:= \frac{10^6 \cdot \log n}{\varepsilon^2} && : \text{ a scaling factor on degree of almost-regular graphs,} \\
\Delta &:= 100s && : \text{ used as a parameter to denote the degree of almost-regular graphs,} \\
F &:= \arg\min_i \left\{ \Delta^{2^i} \geq n^{1/100} \right\} && : \text{ used to bound the number of phases in our algorithm.}
\end{aligned} \tag{6.3}$$

Throughout this section, we typically define the degree of almost-regular graphs by multiplicative factors of s; this is needed to simplify many concentration bounds used in the proofs. We further point out that $F = O(\log \log n)$ and $\Delta^F \in [n^{1/100}, n^{1/50}]$ and hence $\Delta^F = o(\varepsilon)$.

Preprocessing step. The first step in proving Lemma 6.7.1, is to make each connected component G_i of G "more random", i.e., turn it to a graph sampled from $\mathbb{G}$ with larger per-vertex degree. This can be easily done using Lemma 6.6.1 in previous section, as the graph $G_i \sim \mathbb{G}(n_i, 100 \log n)$ has a small mixing time by Proposition 6.3.4 with high probability.

Now consider the following preprocessing process: Recall the parameters defined in Eq (6.3). For $(F \cdot \Delta \cdot s/(100 \log n))$ steps in parallel, we run the algorithm in Lemma 6.6.1 on the original graph G. For each connected component G_i of G, this results us in having F graphs $\widetilde{G}_{i,1}, \ldots, \widetilde{G}_{i,F}$ which are (almost) sampled from the distribution $\mathbb{G}(n_i, \Delta \cdot s)$ (the distribution of these graphs is not exactly identical to this, but is rather close to this distribution in total variation distance which is sufficient for our purpose). As such, we now need to find the connected component of a graph $\widetilde{G}$ which is the union of all $\widetilde{G}_{i,j}$ for i ranging over all connected components of G and $j \in [F]$.

In the following lemma, we design an algorithm for this task. For simplicity of exposition, we state this lemma for the case of finding a spanning tree of one such connected component (i.e., assuming G itself is sampled from $\mathbb{G}$ as opposed to having its connected components sampled from this distribution); however, it would be evident that running this algorithm on the original input results in finding a spanning tree of each connected component separately.

Lemma 6.7.2. *Let $\widetilde{G}$ be a graph on n vertices such that $\widetilde{G} = \widetilde{G}_1 \cup \ldots \cup \widetilde{G}_F$ where $\widetilde{G}_i \sim \mathbb{G}(n, \Delta \cdot s)$. There exists an MPC algorithm that can find a spanning tree of $\widetilde{G}$ with high probability (over both the randomness of the algorithm and the distribution $\mathbb{G}$).*

For any $\delta > 0$, the algorithm can be implemented with $O(n^{1-\delta}) \cdot \text{polylog}(n)$ machines each with $O(n^{\delta}) \cdot \text{polylog}(n)$ memory, and in $O(\frac{1}{\delta} \cdot \log\log n)$ MPC rounds.

We note that in Lemma 6.7.2, the input to the algorithm is the collection of graphs $(\widetilde{G}_1, \ldots, \widetilde{G}_F)$ (i.e., the algorithm knows partitioning of G into its F subgraphs; think of each input edge being labeled by the graph $\widetilde{G}_i$ it belongs to). The rest of this section is devoted to the proof of Lemma 6.7.2. At the end of the section, we use this lemma to prove Lemma 6.7.1. In this section, n always refer to number of vertices in $\widetilde{G}$.

6.7.1. Proof of Lemma 6.7.2: Connectivity on a Single Random Graph

We start by defining a natural operation on graphs in context of connectivity.

Definition 6.2 (Contraction Graph)**.** *For a graph $G(V, E)$ and a partition $\mathcal{C} := \{C_1, \ldots, C_k\}$ of $V(G)$ (not necessarily a component-partition), we construct a* contraction graph *H of G with respect to $\mathcal{C}$ as the following graph:*

1. ***Vertex-set:*** *The vertex-set $V(H)$ of H is a collection of k vertices where $w_i \in V(H)$ is labeled with the component C_i of $\mathcal{C}$, denoted by $C(w_i)$.*
2. ***Edge-set:*** *For any $w \neq z \in V(H)$, there exists an edge $(w, z) \in E(H)$ iff there exists vertices $u \in C(w)$ and $v \in C(z)$ where $(u, v) \in E(G)$ (H contains no parallel edges and no self-loops).*

In other words, H is obtained by "contracting" the vertices of G inside each set of $\mathcal{C}$ into a single vertex and removing parallel edges and self-loops.

Suppose $\mathcal{C}$ is a component-partition of G and H is a contraction graph of G with respect to H. Then it is immediate to see that we can construct a spanning tree (or forest) of G given only spanning trees of each component in $\mathcal{C}$ and a spanning tree of H.

Overview of the algorithm. The algorithm in Lemma 6.7.2 goes through F *phases*. In each phase $i \in [F]$, it only considers the edges in $\widetilde{G}_i$ and use them to "grow" the components of $\widetilde{G}$ found in the previous phases. This part is done using a new *leader-election* algorithm that we design in this chapter. This algorithm takes the contraction graph of $\widetilde{G}_i$ with respect to the set of components found already, and merge these components further to build larger components. The novelty of this leader-election algorithm is that starting from an (almost) d-regular graph, it can grow each component by a factor of (almost) d (as opposed to typical leader-election algorithms that only increase size of each component by a constant factor).

Our main algorithm is then obtained by successively applying this leader election algorithm to contraction graph of $\widetilde{G}_i$ to build relatively large components of $\widetilde{G}_i$ and use them to refine the components found for G. The main step of our proof is to argue that if

contraction graph of $\widetilde{G}_i$ was a *random* (almost) d-regular graph on n' vertices, then the contraction graph of $\widetilde{G}_{i+1}$ in this process would be another random (almost) d^2-regular graph on roughly n'/d vertices. Having achieved this, we can argue that each component of the graph G grows by a quadratic factor in each phase, and hence after only $O(\log\log n)$ phase, each component has size $n^{\Omega(1)}$ (due to technical reasons, one cannot continue this argument until just one connected component of size n remains). Finally, we prove that at this step, the diameter of the remaining graph, i.e., contraction of G on the found components is only $O(1)$. A simple broadcasting algorithm can then be used to found a spanning tree of the remaining graph in $O(1)$ rounds.

A Leader Election Algorithm

We first introduce a simple leader election algorithm, called $\mathsf{LeaderElection}(H, d)$, which gets as an input an (almost) $(d \cdot s)$-regular graph and creates components of size (almost) d in this graph. We note that the description of the algorithm itself does not depend on the fact that H is almost-regular.

$\mathsf{LeaderElection}(H, d)$. A simple leader election algorithm for growing connected components on an (almost) $(d \cdot s)$-regular graph H.

1. Set $L = \emptyset$ initially.
2. For every vertex $v \in V(H)$ **in parallel** independently sample $p(v)$ from the Bernoulli distribution with probability $p := s/d$ and insert u to L iff $p(v) = 1$ (we refer to these vertices as *leaders*).
3. Let $R := V(H) \setminus L$.
4. For any vertex $v \in R$ **in parallel** set $N_L(v)$ be the set of neighbors of v in L in graph H.
5. For any vertex $v \in R$ **in parallel** let $M(v)$ be a vertex $u \in R$ chosen uniformly at random from $N_L(v)$ (we define $M(v) = \perp$ if $N_L(v) = \emptyset$).
6. Return $k := |L|$ sets $S_{v_1}, \ldots, S_{v_k}$ for $v_1, \ldots, v_k \in L$ such that $S_{v_i} = \{v_i\} \cup \{u \in R : M(u) = v_i\}$ (vertices with $M(u) = \perp$ are ignored).

We have the following immediate claim.

Claim 6.7.3. *Suppose in* $\mathsf{LeaderElection}$ *no vertex $v \in R$ has $M(v) = \perp$. Then, the returned collection $S_1, \ldots, S_k$ is a component-partition of H.*

Proof. The induced graph of H on any set S_i contains a star with the leader in S_i being the its center. Hence, each S_i is a component of H. Moreover, by definition, the sets S_i's are disjoint. Finally, since for no vertex $v \in R$, $M(v) = \perp$, S_i's contain all vertices in H. □

The main property of LeaderElection is that when computed on almost regular graphs it results in a component-partition with *almost equal size components*. In other words, if H is a $[\![(1 \pm \varepsilon)d \cdot s]\!]$-almost-regular graph, then the resulting components are of size $[\![(1 \pm O(\varepsilon)) \cdot d]\!]$ each.

Lemma 6.7.4 (Equipartition Lemma)**.** *Let $\overline{\varepsilon} \in (\varepsilon, 1/100)$ and H be a $[\![(1 \pm \overline{\varepsilon})\, d \cdot s]\!]$-almost-regular graph for $d \geq s$. Then, with probability $1 - 1/n^{23}$, for $(S_1, \ldots, S_k) =$ LeaderElection(H, d):*

1. *For all $i \in [k]$, $|S_i| \in [\![(1 \pm 3\overline{\varepsilon})\, d]\!]$,*
2. *$(S_1, \ldots, S_k)$ is a component-partition of $V(H)$.*

Proof. Define $\varepsilon' = \overline{\varepsilon}/10$ and so $s \geq 100 \log n/\varepsilon'^2$ by Eq (6.3). Throughout the proof, we repeatedly use the facts that $[\![(1 \pm \varepsilon')^{-1}]\!] \subseteq [\![(1 \pm 2\varepsilon')]\!]$ and $[\![(1 \pm \varepsilon')^2]\!] \subseteq [\![(1 \pm 3\varepsilon')]\!]$ as $\varepsilon' = o(1)$.

Fix any vertex $u \in R$ and let $d_u \in [\![(1 \pm 10\varepsilon')\, d \cdot s]\!]$ be the degree of u in H. We define d_u random variables $X_1, \ldots, X_{d_u}$ where $X_i = 1$ iff the i-th neighbor of u is chosen as a leader in L and $X_i = 0$ otherwise. Let $X = \sum_i X_i$ denote the number of neighbors of u in L. As the choice of any leader is independent of whether u belongs to L or not, we have $\mathbb{E}[X] = d_u \cdot p \in [\![(1 \pm 10\varepsilon')s]\!]$. Moreover, by Chernoff bound,

$$\begin{aligned}\Pr\left(X \notin [\![(1 \pm \varepsilon')\, \mathbb{E}[X]]\!]\right) &\leq \exp\left(-\frac{\varepsilon'^2 \cdot d_u \cdot p}{2}\right) \leq \exp\left(-\frac{\varepsilon'^2(1 - 10\varepsilon') \cdot s}{2}\right) \\ &\leq \exp\left(-25 \log n\right) \leq \frac{1}{n^{25}}. \quad (\text{as } s \geq 100 \log n/\varepsilon'^2 \text{ and } \varepsilon' = o(1))\end{aligned}$$

Consequently, w.p. $1 - 1/n^{25}$, $|N_L(u)| \in [\![(1 \pm \varepsilon') \cdot (1 \pm 10\varepsilon') \cdot s]\!] \subseteq [\![(1 \pm 12\varepsilon') \cdot s]\!]$ (as $\varepsilon' = o(1)$). By union bound, this event happens for all vertices in R w.p. $1 - 1/n^{24}$. In the following, we condition on this event. The second part of the lemma already follows from this and Claim 6.7.3.

Now fix a vertex $v \in L$. Define $N_R(v)$ as the set of neighbors of v in set R in graph H. The same exact argument as above implies that with probability $1 - 1/n^{24}$, for all vertices in L, $|N_R(v)| \in [\![(1 \pm 12\varepsilon')\, d \cdot s]\!]$. We further condition on this event.

Consider again a vertex $v \in L$. For any vertex $u \in N_R(v)$, we define a random variable Y_u where $Y_u = 1$ iff $M(u) = v$, i.e., u chooses v as its leader. Define $Y = \sum_u Y_u$. We point out that $Y + 1$ is the size of component returned by LeaderElection which contains the leader

v. Hence, it suffices to bound Y to finalize the proof. We have,

$$\mathbb{E}[Y] = \sum_{u \in N_R(v)} \mathbb{E}[Y_u] = \sum_{u \in N_R(v)} \frac{1}{|N_L(u)|} \in [\![\frac{(1 \pm 12\varepsilon')\, d \cdot s}{(1 \pm 12\varepsilon')\, s}]\!] \subseteq [\![\big(1 \pm 25\varepsilon'\big) \cdot d]\!],$$

as $|N_R(v)| \in [\![(1 \pm 12\varepsilon')\, d \cdot s]\!]$ and $|N_L(u)| = [\![(1 \pm 12\varepsilon') \cdot s]\!]$ and $\varepsilon' = o(1)$. By Chernoff bound,

$$\Pr\big(Y \notin [\![\big(1 \pm \varepsilon'\big)\, \mathbb{E}[Y]]\!]\big) \leq \exp\left(-\frac{\varepsilon'^2 \cdot (1 - 25\varepsilon') \cdot d}{2}\right) \leq \exp\left(-25 \log n\right) \leq \frac{1}{n^{25}}.$$

A union bound on all vertices in L implies that $|S_i| \in [\![(1 \pm 27\varepsilon')\, d]\!] \subseteq [\![(1 \pm 30\varepsilon')\, d]\!]$ with probability $1 - 1/n^{24}$. Taking another union bound on all the events conditioned on in the proof, with probability $1 - 1/n^{23}$, we obtain that $|S_i| \in [\![(1 \pm 30\varepsilon')\, d]\!] = [\![(1 \pm 3\overline{\varepsilon})\, d]\!]$, finalizing the proof. □

We have the following claim by the definition of Algorithm LeaderElection.

Claim 6.7.5. *Algorithm* LeaderElection(H, d) *requires* $O(|E(H)|/n^{\delta})$ *machines each with* $O(n^{\delta})$ *memory and* $O(1/\delta)$ *MPC rounds.*

Growing Connected Components

We now use LeaderElection algorithm from the previous section to design our main algorithm which "grows" the size of connected components of G repeatedly over F phases.

GrowComponents$(\widetilde{G}, \Delta)$. An algorithm for "growing" connected components of size up to $n^{\Omega(1)}$ in a given graph $\widetilde{G} = \widetilde{G}_1 \cup \ldots \cup \widetilde{G}_F$ where $\widetilde{G}_i \sim \mathbb{G}(n, \Delta \cdot s)$.

1. Let $\mathcal{C}_1$ be a partition of $V(\widetilde{G})$ into singleton sets.
2. For $i = 1$ to F phases:
   (a) Let $\Delta_i := \Delta^{2^{i-1}}$ and $p_i = \Delta_i^{-1} \cdot s$.
   (b) For every vertex $v \in V(\widetilde{G}_i)$ **in parallel** let $c_i(v) = j$ for $v \in C_j$.
   (c) Construct contraction graph H_i of $\widetilde{G}_i$ (*not* $\widetilde{G}$) with respect to $\mathcal{C}_i$ as follows:
      i. Set H_i to be an empty set inititially.
      ii. For every edge $(u, v) \in E(\widetilde{G}_i)$ **in parallel** add $(C_{c_i(u)}, C_{c_i(v)})$ to H_i.
   (d) Compute $(S_1, \ldots, S_k) =$ LeaderElection(H_i, Δ_i) (hence, each $S_j \subseteq V(H_i)$).
   (e) For each S_j **in parallel** let $C_{i+1,j} = \bigcup_{w \in S_j} C_i(w)$.
   (f) Let $\mathcal{C}_{i+1} = \{C_{i+1,1}, \ldots, C_{i+1,k}\}$.
3. Return the graph H_F.

The following claim is straightforward from GrowComponents and Claim 6.7.5.

Claim 6.7.6. *Algorithm* GrowComponents$(\widetilde{G}, \Delta)$ *requires* $O(|E(\widetilde{G})|/n^{\delta})$ *machines each with* $O(n^{\delta})$ *memory and* $O(F/\delta)$ *MPC rounds.*

We prove that for each phase $i \in [F]$, the contraction graph H_i constructed in this phase is an almost-regular graph with degree roughly $\Delta_i \cdot s$ and discrepancy factor $\varepsilon_i := \big(20^i \cdot \varepsilon\big)$. The following lemma is the heart of the proof.

Lemma 6.7.7. *In* GrowComponents$(\widetilde{G}, \Delta)$*, with high probability, for any* $i \in [F]$*:*

(I) $\mathcal{C}_i$ *is a component-partition of* $\widetilde{G}$ *with* $|C_{i,j}| \in [\![(1 \pm \varepsilon_i)\, \Delta_i/\Delta]\!]$ *for all* $C_{i,j} \in \mathcal{C}_i$*.*

(II) H_i *is a* $[\![(1 \pm \varepsilon_i)\, \Delta_i \cdot s]\!]$*-almost-regular graph on* $n_i \in [\![(1 \pm \varepsilon_i) \cdot n\Delta/\Delta_i]\!]$ *vertices.*

Proof. We prove this lemma inductively.

Base case: $\mathcal{C}_1$ is clearly a component-partition of $\widetilde{G}$ as it only consists of singleton sets and $|C_{1,j}| = 1$ for all $C_{1,j} \in \mathcal{C}_1$. Since $\Delta_1 = \Delta$, this proves the first part of the lemma in the base case. For the second part, as $\mathcal{C}_1$ only consists of singleton sets, $H_1 = \widetilde{G}_1$ and hence $n_1 = n$. Finally, $H_1 = \widetilde{G}_1 \sim \mathbb{G}(n, \Delta \cdot s)$ and hence by Proposition 6.3.2 (as $s \geq 100 \log n/\varepsilon^2$), H_1 is a $[\![(1 \pm \varepsilon)\, \Delta \cdot s]\!]$-almost-regular graph, hence concluding the proof of the base case.

Induction step: Now suppose this is the case for some $i > 1$ and we prove it for $i + 1$. By induction, we have that H_i is a $[\![(1 \pm \varepsilon_i)\, \Delta_i \cdot s]\!]$-almost-regular graph on $n_i \in [\![(1 \pm \varepsilon_i) \cdot n \cdot \Delta/\Delta_i]\!]$ vertices. In this phase, we compute $(S_1, \ldots, S_k) =$ LeaderElection(H_i, Δ_i). We can thus apply Lemma 6.7.4 with parameters $d = \Delta_i$, $p = p_i$, and $\overline{\varepsilon} = \varepsilon_i < 1/100$, and obtain that with high probability,

$$|S_i| \in [\![(1 \pm 3\overline{\varepsilon}) \cdot \Delta_i]\!] = [\![(1 \pm 3\varepsilon_i) \cdot \Delta_i]\!], \tag{6.4}$$

and $(S_1, \ldots, S_k)$ is a component-partition of H_i. In the following, we condition on this event.

Proof of part (I): Since $\mathcal{C}_i$ is a component-partition of $\widetilde{G}$ (by induction), we have that vertices in H_i correspond to components of $\widetilde{G}$, i.e., vertices in $C_i(w)$ for all $w \in V(H_i)$ are connected in $\widetilde{G}$. Moreover, by Lemma 6.7.4, $(S_1, \ldots, S_k)$ is a component-partition of H_i and hence vertices (of H_i) in each S_j for $j \in [k]$ are connected to each other (in H_i). As edges of H_i correspond to edges in $\widetilde{G}_i \subseteq \widetilde{G}$, any $C_{i+1,j} \in \mathcal{C}_{i+1}$ is a component of $\widetilde{G}$, hence $\mathcal{C}_{i+1}$ is a component-partition of $\widetilde{G}$.

We now prove the bound on size of each $C_{i+1,j} \in \mathcal{C}_{i+1}$. By definition,

$$\begin{aligned}
|C_{i+1,j}| &= \sum_{w \in S_j} |C_i(w)| \in [\![|S_j| \cdot (1 \pm \varepsilon_i)\, \Delta_i/\Delta]\!] \quad \text{(by induction hypothesis on } C_i(w) \in \mathcal{C}_i) \\
&\subseteq [\![((1 \pm 3\varepsilon_i) \cdot \Delta_i) \cdot ((1 \pm \varepsilon_i)\, \Delta_i/\Delta)]\!] \qquad \text{(by Eq (6.4))} \\
&\subseteq [\![(1 \pm 5\varepsilon_i) \cdot \Delta_i^2/\Delta]\!] = [\![(1 \pm 5\varepsilon_i) \cdot \Delta_{i+1}/\Delta]\!], \qquad (6.5)
\end{aligned}$$

as $\Delta_i^2 = \Delta_{i+1}$. By the choice of $\varepsilon_{i+1} > 5\varepsilon_i$, this finalizes the proof of the first part. We now consider the second part.

Proof of part (II): Notice that $n_{i+1} = |\mathcal{C}_{i+1}|$ as each set in $\mathcal{C}_{i+1}$ is contracted to a single vertex in H_{i+1}. Since $\mathcal{C}_{i+1}$ partitions $V(G)$, and as by Eq (6.5) each set in $\mathcal{C}_{i+1}$ has size in $[\![(1 \pm 5\varepsilon_i) \cdot \Delta_{i+1}/\Delta]\!]$, we have

$$n_{i+1} \in [\![\frac{n}{(1 \pm 5\varepsilon_i) \cdot \Delta_{i+1}/\Delta}]\!] \subseteq [\![(1 \pm 6\varepsilon_i)\, n \cdot \Delta/\Delta_{i+1}]\!]. \qquad (6.6)$$

As $\varepsilon_{i+1} > 6\varepsilon_i$, this proves the bound on n_{i+1}. It remains to prove H_{i+1} is an $[\![(1 \pm \varepsilon_{i+1})\, \Delta_{i+1} \cdot s]\!]$-almost-regular graph. This is the main part of the argument.

Lemma 6.7.8. *For any vertex $w \in V(H_{i+1})$, degree of w in H_{i+1} is $d_w \in [\![(1 \pm \varepsilon_{i+1})\, \Delta_{i+1} \cdot s]\!]$ with high probability.*

Proof. Recall that H_{i+1} is a contraction graph of $\widetilde{G}_{i+1}$ with respect to the partition $\mathcal{C}_{i+1}$. We define $C = C_{i+1}(w) \in \mathcal{C}_{i+1}$. In H_{i+1}, w has an edge to another vertex $z \in V(H_{i+1})$ iff there exists a vertex $u \in C \subseteq V(\widetilde{G}_{i+1})$ such that u has an edge to some vertex $v \in C_{i+1}(z)$ in the graph $\widetilde{G}_{i+1}$. As such, degree of w is equal to the number of sets $C_{i+1,j} \subseteq V(\widetilde{G}_{i+1})$ such that there is an edge $(u, v) \in E(\widetilde{G}_{i+1})$ for $u \in C$ and $v \in C_{i+1,j}$.

Now consider the process of generating $\widetilde{G}_{i+1} \sim \mathbb{G}(n, \Delta \cdot s)$ and notice that the edges chosen in $\widetilde{G}_{i+1}$ are chosen independent of the choice of $\mathcal{C}_{i+1}$ as $\mathcal{C}_{i+1}$ is only a function of the graphs $\widetilde{G}_1, \ldots, \widetilde{G}_i$. Moreover, recall that in $\mathbb{G}(n, \Delta \cdot s)$ each vertex chooses $\Delta \cdot s/2$ other vertices uniformly at random to connect to (and then we remove the direction of edges). For any two sets $S, T \subseteq V(\widetilde{G}_{i+1})$, we say that S "hits" T if there exists a vertex in S which picks a directed edge to some vertex in T in the process of generating $\widetilde{G}_{i+1}$ (so it is possible that S hits T but T does not hit S). Let $K \subseteq [k]$ be such that for each $j \in K$, either C hits $C_{i+1,j}$ or vice versa. By the above argument $d_w \in [\![|K| \pm 1]\!]$ (to account for the fact that C hitting C does not change the degree of w as we have no self-loops in H_{i+1}). In the following two claims, we bound $|K|$.

Claim 6.7.9. *Let $K^+ \subseteq [k]$ be the set of all indices $j \in [k]$ such that C hits $C_{i+1,j}$. Then, with high probability, $|K^+| \in [\![(1 \pm \varepsilon_{i+1})\, \Delta_{i+1} \cdot s/2]\!]$.*

Proof. We model the number of sets hit by C as a balls and bins experiment (see Section 2.1.2): "balls" are the edges going out of vertices in C in construction of $\widetilde{G}_{i+1}$ in $\mathbb{G}$ and "bins" are the sets $C_{i+1,j}$ for $j \in [k]$. Hence, non-empty bins are exactly the set K^+ and thus it suffices to bound number of non-empty bins.

In $\widetilde{G}_{i+1}$, any vertex in C is choosing $\Delta \cdot s/2$ directed edges. As such, the total number of balls in this argument is $N = |C| \cdot \Delta \cdot s/2 \in [\![(1 \pm 5\varepsilon_i)\, \Delta_{i+1} \cdot s/2]\!]$ by the bound proven on $|C|$ in Eq (6.5).

The total number of bins in this argument is $B = |\mathcal{C}_{i+1}| = n_{i+1} \in [\![(1 \pm 6\varepsilon_i)\, n \cdot \Delta/\Delta_{i+1}]\!]$ as proven in Eq (6.6). Moreover, for any $j \in [k]$, $|C_{i+1,j}| \in [\![(1 \pm 5\varepsilon_i)\, \Delta_{i+1}/\Delta]\!]$ (as stated above for C). As such, the ratio between the largest and smallest set in $\mathcal{C}_{i+1}$ is in $[\![(1 \pm 10\varepsilon_i)]\!]$. Also, edges going out of C are chosen uniformly at random from, and hence each bin in this argument is chosen with probability in $[\![(1 \pm 10\varepsilon_i) \cdot B^{-1}]\!]$. Moreover,

$$\frac{B}{N} \in [\![\frac{(1 \pm 6\varepsilon_i)\, n \cdot \Delta/\Delta_{i+1}}{(1 \pm 5\varepsilon_i)\, \Delta_{i+1} \cdot s/2}]\!] \Longrightarrow \frac{B}{N} \geq \frac{n \cdot \Delta}{2\Delta_F^2} \gg \text{polylog}(n) \gg \frac{1}{10\varepsilon_i},$$

where the inequalities are by choice of F and ε because $\varepsilon_F = o(1)$ and $\Delta_F = \Delta^{2^F} \leq n^{1/50}$.

Let X be the number of non-empty bins in this process. By Proposition 2.1.5 for this balls and bins experiment: $\Pr\left(X \notin [\![(1 \pm 20\varepsilon_i) \cdot N]\!]\right) \leq \exp\left(-\frac{100\varepsilon_i^2 \cdot N}{2}\right) = 1/n^{\omega(1)}$. Hence, with high probability, the total number of non-empty bins is in $[\![(1 \pm 20\varepsilon_i)\, \Delta_{i+1} \cdot s/2]\!]$, which finalizes the proof as $\varepsilon_{i+1} = 20\varepsilon_i$. □ Claim 6.7.9

An interpretation of Claim 6.7.9 is that that distribution of H_{i+1} is $\mathbb{G}(n_{i+1}, \Delta_{i+1} \cdot s)$ with the difference that the number of out-edges chosen in $\mathbb{G}$ is not exactly $\Delta_{i+1} \cdot s$ (but quite close to it for each vertex). As such, we would expect H_{i+1} to still behave similarly as $\mathbb{G}(n_{i+1}, \Delta_{i+1} \cdot s)$; in particular, be almost-regular with high probability. The following claim is analog of Proposition 6.3.2 for the distribution of H_{i+1}.

Claim 6.7.10. *Let $K^- \subseteq [k]$ be the set of all indices $j \in [k] \setminus K^+$ such that $C_{i+1,j}$ hits C. Then, with high probability, $|K^-| \in [\![(1 \pm \varepsilon_{i+1})\, \Delta_{i+1} \cdot s/2]\!]$.*

Proof. Fix $j \in [k] \setminus K^+$. As shown in Claim 6.7.9, $c_j := |C_{i+1,j}| \cdot \Delta \cdot s/2 \in [\![(1 \pm 5\varepsilon_i)\, \Delta_{i+1} \cdot s/2]\!]$ edges are going out of vertices in $C_{i+1,j}$. Any such edge, would hit the set C with probability $p := |C|/n$. Let $\varepsilon' = (1/\log n)^{10} \ll \varepsilon$. We have,

$$\Pr\left(C_{i+1,j} \text{ hits } C\right) = 1 - (1-p)^{c_j} \in [\![(1 \pm c_j p) \cdot c_j p]\!] \subseteq [\![(1 \pm \varepsilon') \cdot c_j p]\!].$$
$$\text{(as } (1-x) \leq e^{-x} \leq 1 - x + x^2 \text{ and } c_j \cdot p = \widetilde{O}(\Delta_F^2)/n = \widetilde{O}(n^{2/50})/n \ll \varepsilon')$$

Moreover, we have $k = n_{i+1} \in [\![(1 \pm 6\varepsilon_i)\, n \cdot \Delta/\Delta_{i+1}]\!]$ and $|K^+| = [\![(1 \pm \varepsilon_{i+1})\, \Delta_{i+1} \cdot s/2]\!]$

and since $\widetilde{O}(\Delta_F^2)/n \ll \varepsilon'$, we have that $|[k] \setminus K^+| \in [\![(1 \pm \varepsilon') \cdot |k|]\!]$.

Let $X = |K^-|$ denotes the number of sets $C_{i+1,j} \in [k] \setminus K^+$ that hit C. Consequently,

$$\begin{aligned}\mathbb{E}[X] &\in [\![(1 \pm \varepsilon') \cdot \sum_{j \in [k] \setminus K^+} c_j p]\!] \subseteq [\![(1 \pm \varepsilon') \cdot (1 \pm \varepsilon') \cdot |k| \cdot ((1 \pm 6 \cdot \varepsilon_{i+1}) \Delta_{i+1})^2 \cdot s/2\Delta n]\!] \\ &\qquad \text{(as } p = |C|/n \text{ and } |C| \in [\![(1 \pm 5\varepsilon_i) \Delta_{i+1}/\Delta]\!] \text{ and } c_j \in [\![(1 \pm 5\varepsilon_i) \Delta_{i+1} \cdot s/2]\!]) \\ &\subseteq [\![(1 \pm 3\varepsilon') \cdot (1 \pm 13 \cdot \varepsilon_i) \Delta_{i+1} \cdot s/2]\!].\end{aligned}$$

Finally, X is a sum of independent random variables and hence by Chernoff bound,

$$\Pr(X \notin [\![(1 \pm \varepsilon_i) \mathbb{E}[X]]\!]) \leq \exp\left(-\frac{\varepsilon^2 \cdot s}{4}\right) \leq \frac{1}{n^{25}}.$$

Thus, w.h.p. $X \in [\![(1 \pm 15\varepsilon_i) \Delta_{i+1} \cdot s/2]\!]$. As $\varepsilon_{i+1} > 15\varepsilon_i$, we are done. □ Claim 6.7.10

Lemma 6.7.8 now follows immediately from Claims 6.7.9 and 6.7.10. □ Lemma 6.7.8

To conclude the proof of Lemma 6.7.7, we simply take a union bound on all vertices $w \in V(H_{i+1})$ and by Lemma 6.7.8 obtain that with high probability $d_w \in [\![(1 \pm \varepsilon_{i+1}) \Delta_{i+1} \cdot s]\!]$. This implies that H_{i+1} is a $[\![(1 \pm \varepsilon_{i+1}) \Delta_{i+1} \cdot s]\!]$-almost regular graph. □ Lemma 6.7.7

We also state the following corollary of Lemma 6.7.7 which states that each graph H_i is sampled from a distribution which is in spirit of $\mathbb{G}(n_i, \Delta_i \cdot s)$ (with additional "noise").

Proposition 6.7.11. *With high probability, distribution of each graph H_i in* GrowComponents *is a graph on $n_i \in [\![(1 \pm \varepsilon_i) \cdot n\Delta/\Delta_i]\!]$ vertices in which we first pick $[\![(1 \pm \varepsilon_i) \Delta_i \cdot s/2]\!]$ many neighbors for each vertex where the other endpoint is chosen with probability in $[\![(1 \pm 2\varepsilon_i) \cdot n_i^{-1}]\!]$ and then remove the direction of edges.*

The proof of this proposition is identical to the proof of Claim 6.7.9 using the fact that H_i is almost-regular by Lemma 6.7.7 (see also the discussion after Claim 6.7.9).

Finally, we claim that GrowComponents also finds a spanning tree of components in $\mathcal{C}_F$.

Claim 6.7.12. *Let T be the set of edges chosen in executions of* LeaderElection *(in defining $M(\cdot)$ for each non-leader vertex) in the course of execution of* GrowComponents$(\widetilde{G}, \Delta)$. *With high probability, the induced subgraph of T on each component in $\mathcal{C}_F$ is a spanning tree.*

Proof. Follows immediately from Claim 6.7.3 and the fact that each $\mathcal{C}_{i+1}$ is formed by merging already found components of $\widetilde{G}$ (see also the discussion after Definition 6.2). □

Building the Spanning Tree

Recall that by the choice of F, $\Delta_F \in [n^{1/100}, n^{1/50}]$. By running GrowComponents$(\widetilde{G}, \Delta)$, we obtain a graph H_F which consists of $n_F \in [\![(1 \pm o(1)) \cdot n \cdot \Delta/\Delta_F]\!]$ vertices (as $\varepsilon_F = o(1)$ by Eq (6.3)). Additionally, by Proposition 6.7.11, with high probability, H_F is a "random" graph (in spirit of $\mathbb{G}$) with $[\![(1 \pm o(1))\,\Delta_F]\!]$ "out-degree" before removing the direction of edges. We use this to bound the diameter of H_F.

Claim 6.7.13. *Diameter of H_F is $D = O(1)$ with high probability.*

Proof. We condition on the event that distribution of H_F is as stated in Proposition 6.7.11. We argue that for any set $S \subseteq V(H_F)$, the neighborset $N(S)$ of S in H_F has size

$$|N(S)| \geq \min\left\{2n_F/3, \Delta_F \cdot |S|\,/20\right\}.$$

The proof of this claim is exactly as in proof of Proposition 6.3.4, using the analogy between $\mathbb{G}$ and distribution of H_F (with a minor additional care to account for the "noise" in H_F). We omit the details of this proof.

By the above equation, the k-hop neighborhood of any vertex in H_F contains either at least $2n_F/3$ or $(\Delta_F/20)^k$ vertices. In particular, for $k' = \log_{(\Delta_F/20)} n$, the k'-hop neighborhood of any vertex contains at least $2n_F/3$ vertices. This implies that the $2k'$-hop neighborhood of any vertex contains the whole graph, hence the diameter of H_F is $O(k')$. Since $\Delta_F \in [n^{1/100}, n^{1/50}]$, we obtain that diameter of H_F is $O(1)$. □

We use the above claim to design a very simple algorithm to build a spanning tree of H_F. We then combine this algorithm with GrowComponents to prove Lemma 6.7.2.

Claim 6.7.14. *Let H be any graph with m edges, n vertices, and diameter D. A spanning tree of H can be found in $O(D/\delta)$ MPC rounds with $O(m^{1-\delta})$ machines with memory $O(m^{\delta})$ for any $\delta > 0$.*

Proof. We pick any arbitrary vertex $v \in H$. The algorithm proceeds in D iterations. In the first iteration, v informs all its neighbors in H and add these edges to the underlying spanning tree. In the next iteration, the neighbors of v inform all their neighbors; any vertex informed which has already not chosen an edge in the spanning tree would pick one of its incoming edges and add it to the spanning tree. We continue like this until after D iterations all vertices have a neighboring edge in the spanning tree.

It is straightforward that one can implement this algorithm in $O(D/\delta)$ MPC rounds on machines of memory $O(m^{\delta})$, hence finalizing the proof. □

We are now ready to conclude the proof of Lemma 6.7.2.

Proof of Lemma 6.7.2. By Claim 6.7.12, we can find a spanning tree of every component of $\widetilde{G}$ in $\mathcal{C}_F$. This step requires $O(\log\log n/\delta)$ MPC rounds on $\widetilde{O}(n^{1-\delta})$ machines of memory $\widetilde{O}(n^{\delta})$ by Claim 6.7.6 (as $\left|E(\widetilde{G})\right| = \widetilde{O}(n)$ by construction). Note that the components in $\mathcal{C}_F$ correspond to vertices of H_F and hence by finding a spanning tree of H_F we obtain a spanning tree of $\widetilde{G}$.

By Claim 6.7.13, diameter of H_F is only $O(1)$. Hence, by the algorithm in Claim 6.7.14, we can find a spanning tree of H_F in only $O(1/\delta)$ rounds on machines of memory $O(n^{\delta})$. Combining these trees, we obtain a spanning tree of $\widetilde{G}$, finalizing the proof. □

6.7.2. Proof of Lemma 6.7.1: Connectivity in a Disjoint Union of Random Graphs

Proof of Lemma 6.7.1. We perform the preprocessing step introduced in the beginning of the section to create the graph $\widetilde{G}$ that is a graph which consists of F copies of $\mathbb{G}(n_i, \Delta \cdot s)$ where n_i is the number of vertices in the connected component G_i of G. By Proposition 6.3.4, with high probability every connected component of G which is sampled from $\mathbb{G}$ (with degree $100\log n$) has mixing time of $\text{polylog}(n)$. We can thus apply Lemma 6.6.1 to implement this step using $\widetilde{O}(n^{1-\delta})$ machines and $\widetilde{O}(n^{\delta})$ memory per machine in $O(\log\log n/\delta)$ rounds.

We then run the algorithm in Lemma 6.7.2 on the whole graph. We can now analyze the algorithm in Lemma 6.7.2 on the set of vertices belonging to each connected component $\widetilde{G}_i$ of $\widetilde{G}$ separately. It is immediate to verify that performance of algorithm in Lemma 6.7.2 on each $\widetilde{G}_i$ is only a function of $\widetilde{G}_i$ and hence the correctness of the algorithm follows exactly as in Lemma 6.7.2. Hence, in $O(\log\log n/\delta)$ rounds, with high probability, we obtain a spanning tree of each $\widetilde{G}_i$. We then assign a unique label to each spanning tree found and mark the vertices based on which spanning tree they belong to. Each label now corresponds to $V(\widetilde{G}_i) = V(G_i)$, hence we can identify all connected components of G, finalizing the proof. The bound on the memory requirement and number of machines follows from Lemma 6.7.2. □

6.8. Putting Everything Together

We now put all components of our algorithms in the previous three sections together and prove the following theorem which formalize Result 6.1 in Section 6.1.

Theorem 6.5. *There exists a randomized MPC algorithm that with high probability identifies all connected components of any given undirected n-vertex graph $G(V,E)$ with m edges and a lower bound of $\lambda \in (0,1)$ on the spectral gap of any of its connected components.*

For any $\delta > 0$, the algorithm can be implemented with $O(\frac{1}{\lambda^2} \cdot m^{1-\delta} \cdot \text{polylog}(n))$ machines each with $O(m^{\delta} \cdot \text{polylog}(n))$ memory, and in $O(\frac{1}{\delta} \cdot (\log\log n + \log(1/\lambda)))$ MPC rounds.

Proof. We prove this theorem by applying the transformation steps to G.

Step 1. Let $G_1 := G$ with $n_1 := |V(G_1)|$ and $m_1 := |E(G_1)|$. We apply Lemma 6.5.1 to G_1 to obtain a Δ-regular graph G_2 with the following properties (with high probability): there is a one-to-one correspondence between connected components of G_1 and G_2, and each connected component of G_2 has mixing time $T_{\gamma^*} = O(\log n/\lambda)$ with $\gamma^* = n^{-10}$. Moreover, $n_2 := |V(G_2)| = O(m_1)$ and $\Delta = O(1)$. By identifying connected components of G_2, we immediately identify connected components of G_1.

This step can be implemented in $O(m^{1-\delta})$ machines with $O(m^{\delta})$ memory in $O(1/\delta)$ MPC rounds by Lemma 6.5.1 (as $m_1 = m$).

Step 2. We apply Lemma 6.6.1 to G_2 with $T = T_{\gamma^*}$ to (with high probability) obtain a graph G_3 such that $V(G_2) = V(G_3)$ and for any connected component $G_{2,i}$ on $n_{2,i}$ vertices, the induced subgraph of G_3 on vertices $V(G_{2,i})$, denoted by $G_{3,i}$, is a connected component of G_3 with distribution $\textsc{dist}(G_{3,i})$ where $|\textsc{dist}(G_{3,i}) - \mathbb{G}(n_{2,i}, 100\log n)|_{tvd} \leq n^{-10}$. Finding connected components of G_2 is equal to finding connected components of G_3.

This step can be implemented with $\widetilde{O}(n_2^{1-\delta})$ machines with $\widetilde{O}(n_2^{\delta})$ memory in $O(\log T/\delta)$ rounds by Lemma 6.6.1. Plugging in these parameters, this step is implementable with $\widetilde{O}(m^{1-\delta})$ machines with $\widetilde{O}(m^{\delta})$ memory in $O(\frac{1}{\delta}(\log\log n + \log(1/\lambda)))$ rounds.

Step 3. Let $n_3 = n_2$ be the number of vertices in G_3. We apply Lemma 6.7.1 to G_3 to identify the connected components of G. The distribution of each connected component $G_{3,i}$ of G_3 is (n^{-8})-close in total variation distance to $\mathbb{G}(n_{2,i}, 100\log n)$ ($n_{2,i} = |V(G_{3,i})|$). Hence, by the guarantee of Lemma 6.7.1 and Fact 2.6.7, with high probability we are able to identify connected components of the graph G_3. This allows us to identify connected components of G_2 and in turn $G_1 = G$.

This step can be implemented in $\widetilde{O}(n_3^{1-\delta})$ machines with $\widetilde{O}(n_3^{\delta})$ memory in $O(\log\log n_3/\delta)$ rounds by Lemma 6.7.1. Plugging in the value of these parameters, we obtain that this step is implementable with $\widetilde{O}(m^{1-\delta})$ machines with $\widetilde{O}(m^{\delta})$ memory in $O(\log\log n/\delta)$ MPC rounds. This concludes the proof of the theorem. □

Extension to Unknown Spectral Gaps

A simple modification of our algorithm in Theorem 6.5 allows for implementing it without having a prior knowledge of the spectral gap of each underlying connected component at the cost of slightly worse parameters.

Corollary 6.6. *There exists a randomized MPC algorithm that for any $\delta > 0$, with high probability identifies all connected components of any given undirected n-vertex graph $G(V, E)$ with m edges such that any connected component G_i with spectral gap $\lambda_2(G_i)$ (unknown to the algorithm) would be identified by the algorithm after*

$$O(\frac{1}{\delta} \cdot \left(\log\log n \cdot \log\log\left(\frac{1}{\lambda_2(G_i)}\right) + \log\left(\frac{1}{\lambda_2(G_i)}\right)\right))$$

MPC rounds.

The algorithm requires $O(\frac{1}{\lambda^{2.1}} \cdot m^{1-\delta} \cdot \text{polylog}(n))$ machines with $O(m^{\delta} \cdot \text{polylog}(n))$ memory, where $\lambda := \min_i \lambda_2(G_i)$ is the minimum spectral gap of connected components of G.

Proof. We first choose $\lambda'_1 = 1/2$ and run the algorithm in Theorem 6.5 on G with this choice of λ'. Let $\mathcal{C} := \{C_1, \ldots, C_k\}$ be the sets identified as connected components of G by this algorithm. We note that algorithm in Theorem 6.5 would always return a component-partition of $V(G)$ and hence if u and v belong to some $C_i \in \mathcal{C}$, u and v also belong to the same connected component in G. However, it is possible that there exists some u and v such that $u, v \in G_i$ (for some particular connected component) but $C(u) \neq C(v)$ (as λ' is not necessarily as small as spectral gap of G_i). It is easy to see that without loss of generality we can assume such u and v are neighbors in G. Hence, we can run a simple post-processing step to mark all components in $\mathcal{C}$ which are a strict subset of some connected component of G, i.e., are "growable", and return the remaining components as connected components of G. This step can be implemented in $O(1/\delta)$ MPC rounds on machines of memory $O(m^{\delta})$.

We then recursively perform the above procedure by setting $\lambda'_j = (\lambda'_{j-1})^{1.1}$ in j-th recursion step on the vertices in marked components. Fix any connected component G_i of G. It is immediate that whenever $\lambda'_j \leq \lambda_2(G_i)$, the above procedure return this connected component (and hence would not mark it further). This means that after $j^\star = O(\log\log(\frac{1}{\lambda_2(G_i)}))$ recursion steps, we have $\lambda'_{j^\star} \leq \lambda_2(G_i)$ and hence G_i would be returned as a connected component. The total number of MPC rounds in these recursion steps is at most

$$\begin{aligned} O(\frac{1}{\delta}) \cdot \sum_{j=1}^{j^\star} \left(\log\log n + \log\frac{1}{\lambda'_j}\right) &= O(\frac{1}{\delta}) \cdot \left(\log\log n \cdot j^\star + \sum_{j=1}^{j^\star} (1.1)^j\right) \\ &= O(\frac{1}{\delta} \cdot \left(\log\log n \cdot \log\log\left(\frac{1}{\lambda_2(G_i)}\right) + \log\left(\frac{1}{\lambda_2(G_i)}\right)\right)). \end{aligned}$$

Finally, it is easy to see that by the time G_i is output, the algorithm has used $\widetilde{O}(\frac{1}{(\lambda'_{j^*})^2} m^{1-\delta})$ machines, each with $\widetilde{O}(m^{\delta})$ memory; as $\lambda'_{j^*} \geq \lambda_2(G_i)^{1.1}$, we obtain the final result. □

6.9. A Mildly Sublinear Space Algorithm for Connectivity

We now present a simple algorithm for solving the sparse connectivity problem (in general, e.g., with no assumption on spectral gap, etc.) using $o(n)$ memory per-machine,.

Theorem (Restatement of Theorem 6.2)**.** *There exists an MPC algorithm that given any arbitrary graph $G(V, E)$ with high probability identifies all connected components of G in $O(\log\log n + \log\left(\frac{n}{s}\right))$ MPC rounds on machines of memory $s = n^{\Omega(1)}$.*

As a corollary of Theorem 6.2, $O(\log\log n)$ rounds suffice to solve the connectivity problem even when memory per-machine is mildly sublinear, i.e., is $O(n/\text{polylog}(n))$, and that as long as the memory per machine is $n^{1-o(1)}$, we can always improve upon the $O(\log n)$-round classical PRAM algorithms for the connectivity problem on any arbitrary graph.

The algorithm in Theorem 6.2 is a simple application of the toolkit we developed for proving our main result in Theorem 6.5, combined with the linear-sketching algorithm of Ahn *et al.* [10] for graph connectivity. In particular, we use the following result from [10].

Proposition 6.9.1 ([10])**.** *Let H be any graph partitioned between $|V(H)|$ players such that each player receives all edges incident on a unique vertex in $V(H)$ (hence each edge is received by exactly two players). There exists a randomized algorithm in which every player sends a message of size $O(\log^3 |V(H)|)$ bits to a central coordinator who can output all connected components of H using only these messages with high probability. The algorithm requires players to have access to* $\text{polylog}(|V(H)|)$ shared random bits.

We are now ready to present the algorithm in Theorem 6.2. We shall emphasize that unlike in our main result in Theorem 6.5, to prove Theorem 6.2, we do not need the full power of essentially any of the steps we developed earlier in the chapter and this result can be achieved using much simpler techniques as we show below.

SublinearConn(G). A mildly sublinear space algorithm for connectivity on a given graph.

1. Set $d := \frac{n\cdot(\log n)^4}{s}$ and $t := \left(d^3 \cdot 100\log n\right)$, and run SimpleRandomWalk(G, t).
2. Create a graph $\widetilde{G}$ from G by connecting every vertex $v \in V(G)$ to all *distinct* vertices visited in the random walk starting from v computed in the previous step.
3. Run LeaderElection($\widetilde{G}, d$) and let H be the graph obtained by contracting any component found by LeaderElection to a single vertex.
4. Remove self-loops and duplicate edges from H and run the algorithm in Proposition 6.9.1 on H by using a dedicated machine to simulate a single player.

We are now ready to prove Theorem 6.2 using the SublinearConn algorithm.

Proof of Theorem 6.2. The correctness is based on the following observations:

1. Even though G is *not* a regular graph (as is needed in the statement of Theorem 6.4), SimpleRandomWalk(G, t) still finds a random walk of length t from every vertex (by the discussions before Observation 6.6.2 and Lemma 6.6.6). These walks are however *not* independent of each other but we shall not need this property. We further note that to actually find all vertices in the walk (and not only the final vertex) we use the Mark procedure defined previously.
2. A random walk of length $O(d^3 \log n)$ from any vertex would either visit all vertices in its connected component or at least d distinct vertices with high probability. This follows from a conjecture of Linial proven by Barnes and Feige in [47] that states that the expected time to visit N distinct vertices by a random walk is $O(N^3)$.
3. It follows from the previous part that the minimum degree of graph $\widetilde{G}$ is at least d with high probability. Even though $\widetilde{G}$ is *not* an almost-regular graph, it follows immediately from Claim 6.7.3 and the proof of second part of Lemma 6.7.4 that components found by LeaderElection contain all vertices of $\widetilde{G}$ with high probability.
4. It follows from the previous part that $|V(H)| = O(n \log n/d) = O(s/\log^3 n)$ with high probability (we set sampling probability of leaders in LeaderElection to $\Theta(\log n/d)$ for this part as this parameter is already enough for the previous argument to work). The correctness now follows from Proposition 6.9.1, as vertices in H are components of G.

To bound the number of rounds, we need $O(\log t) = O(\log \log n + \log\left(\frac{n}{s}\right))$ to implement SimpleRandomWalk(G, t) by Claim 6.6.7, $O(1)$ rounds for LeaderElection by Claim 6.7.5, and $O(1)$ rounds for final step by Proposition 6.9.1 and the fact that we can share polylog(n) random bits as well as removing duplicate edges in $O(1)$ rounds on machines with memory $n^{\Omega(1)}$. To bound the memory, we need $n^{\Omega(1)}$ memory to implement SimpleRandomWalk and LeaderElection and $O(|V(H)| \cdot \log^3(n))$ to implement the final step by Proposition 6.9.1. As argued, $|V(H)| = O(s/\log^3(n))$ and hence $O(s)$ memory is sufficient here. □ Theorem 6.2

6.10. A Lower Bound on Well-Connected Graphs

Our algorithmic results in this chapter suggested that sparse connectivity is potentially "much simpler" on graphs with moderate expansion (i.e., with spectral gap $\lambda \geq 1/\text{polylog}(n)$) than on typical graphs. It is then natural, although perhaps *too* optimistic, to wonder whether sparse connectivity on well-connected graphs is at all "hard" or not; for example, can we achieve an $O(\log \log n)$-round algorithm for finding well-connected components of a graph using only polylog(n) memory per machine in the MPC model, or perhaps directly

in the PRAM model? Such a possibility would indeed imply that one does not need the "full power" of MPC algorithms (more local storage and computational power) to solve the sparse connectivity problem even on well-connected graphs. As we prove next, this is indeed not the case. Our lower bound hence supports the main message of our work on achieving *truly improved* algorithms in the MPC model using the full power of this model.

We prove an *unconditional* lower bound on the number of MPC rounds required to solve the connectivity problem on *sparse* undirected graphs with a *constant spectral gap*. More formally, we henceforth denote by $\mathsf{ExpanderConn}_n$ the decision *promise* problem of determining connectivity on n-vertex graphs G, where in both cases (each connected component of) G is guaranteed to be a sparse expander ($|E(G)| = O(n)$ and the spectral gap of each component is $\lambda_2 = \Omega(1)$).

Theorem 6.7 (Lower bound for expander connectivity)**.** *Every MPC algorithm for the problem of* $\mathsf{ExpanderConn}_n$ *with s space per machine (and an* arbitrary *number of machines) requires $r = \Omega(\log_s n)$ rounds of computation. This holds even against randomized MPC protocols with constant error probability.*

Theorem 6.7, for example, suggests that any MPC algorithm with $\text{polylog}(n)$ memory per machine for connectivity even on union of expanders requires $\Omega(\log n/\log\log n)$ MPC rounds. In Remark 6.10.5, we also show a similar situation for (EREW) PRAM algorithms.

We remark that by a result of [277], the lower bound in Theorem 6.7 is asymptotically the best possible unconditional lower bound short of proving that $\mathbf{NC}^1 \subsetneq \mathbf{P}$ which would be a major breakthrough in complexity theory.

Our lower bound is an adaptation of the argument in [277], who showed the same (asymptotic) lower bound for any (nontrivial) monotone graph property, albeit without the spectral gap nor the sparsity restrictions. They prove a general relationship between the round complexity of an MPC algorithm for computing a function f and the *approximate degree* of f (see Theorem 3.5 and Proposition 2.7 in [277]). More formally, for a Boolean function $f : \{0,1\}^n \to \{0,1\}$, let

$$\widetilde{deg}_{\varepsilon}(f) := \min_{P:\{0,1\}^n \to \mathbb{R}} \{deg(P) \mid |f(x) - P(x)| \leq \varepsilon \ \forall\ x \in \{0,1\}^n\}$$

denote the *ε-approximate degree*, i.e., the lowest degree of an (n-variate) *real* polynomial that uniformly ε-approximates f on the hypercube. The following proposition then follows from Corollaries 3.6 and 3.8 and Proposition 2.7 in [277].

Proposition 6.10.1 ([277])**.** *If $f : \{0,1\}^n \to \{0,1\}$ is computable by an r-round randomized ε-error MPC algorithm with space s per machine, then $\widetilde{deg}_{\varepsilon}(f) \leq s^{\Theta(r)}$.*

By Proposition 6.10.1, proving Theorem 6.7 boils down to showing $\widetilde{deg}_{\varepsilon}(\mathsf{ExpanderConn}_n) = n^{\Omega(1)}$, as this would imply an $r = \Omega(\log_s n)$ round lower bound for MPC algorithms for $\mathsf{ExpanderConn}_n$ with s memory per machine.

To prove such lower bounds on approximate degree of functions, [277] further observed that it suffices to lower bound the *deterministic decision tree* complexity $DT(f)$ (i.e., the *query complexity*) of the underlying function, as it is known to imply a polynomially-related lower bound on its approximate degree.

Proposition 6.10.2 (Decision-tree complexity vs. approximate polynomial degree, [52, 260])**.** *For any Boolean function $f : \{0,1\}^n \to \{0,1\}$, it holds that $\widetilde{deg}_{1/3}(f) \geq \Omega\left(DT(f)^{1/6}\right)$.*

We remark that the same bound applies to *partial* functions defined on $\mathcal{D} \subseteq \{0,1\}^n$, using a straightforward generalization of the block-sensitivity measure and approximate degrees (see, e.g., Theorem 4.13 and the comment following it in [52]). It therefore remains to prove a lower bound on the deterministic decision tree complexity of $\mathsf{ExpanderConn}_n$, which is the content of the next lemma.

Lemma 6.10.3. $DT(\mathsf{ExpanderConn}_n) = \Omega(n/\log n)$.

We shall use the following claim to construct our hard instances in the proof the lower bound in Lemma 6.10.3.

Claim 6.10.4. *There exists a collection of $k = \Omega(n)$ graphs $\mathcal{B} := B_1, \ldots, B_k$ on the same set V of n vertices such that:*

1. *Each B_i is a d-regular graph with some fixed $d = O(1)$ and has spectral gap $\lambda_2(B_i) = \Omega(1)$.*
2. *Any edge $e \in V \times V$ appears in at most $O(\log n)$ different graphs $B_i \in \mathcal{B}$.*

Proof. We prove this claim using a probabilistic argument. Fix $d = 100$. Recall the definition of distribution $\mathcal{G}_{n,d}$ from Section 6.5, i.e., the uniform distribution on d-regular graphs on n vertices. We pick $k = n/100d$ graphs $\mathcal{B} := B_1, \ldots, B_k$ independently from $\mathcal{G}_{n,d}$. By Corollary 6.3 and a union bound, with high probability all these graphs are expanders with $\lambda_2(B_i) = \Omega(1)$.

Now consider any fixed edge $e \in V \times V$. For $i \in [k]$, define indicator random variables $X_i \in \{0,1\}$ where $X_i = 1$ iff $e \in B_i$. Define $X := \sum_{i=1}^{k} X_i$ as the total number of graphs in $\mathcal{B}$ to which e belongs to. We have, $\mathbb{E}[X] = \sum_{i=1}^{k} \mathbb{E}[X_i] = k \cdot \Pr_{B \sim \mathcal{G}_{n,d}}(e \in B) = k \cdot \frac{2nd}{n^2} = \frac{1}{100}$.

As such, by Chernoff bound, the probability that $X \geq 4 \log n$, i.e., e appears in more than $4 \log n$ graphs is at most $1/n^3$. Taking a union bound on all edges $e \in V \times V$, implies

that with high probability no edge appears in more than $O(\log n)$ different graphs in $\mathcal{B}$.

Taking a union bound on the two events above, we obtain that with high probability, $\mathcal{B}$ satisfies the requirement of the claim. This implies that in particular there should exists such collection $\mathcal{B}$, finalizing the proof. □ Claim 6.10.4

We now present the proof of Lemma 6.10.3, which completes the proof of Theorem 6.7.

Proof of of Lemma 6.10.3. Let S, T be two disjoint subsets of vertices of size $n/2$ each and G_S and G_T be two d-regular expanders with $\lambda_2 = \Omega(1)$ on vertices S and T, respectively. Moreover, Let $\mathcal{B} := \{B_i\}_{i=1}^{k}$ be the collection of k expanders in Claim 6.10.4. Note that the collection $\mathcal{B} = \{B_1, \ldots, B_k\}$ is fixed in advance and known to the query algorithm (or equivalently the decision tree). Our final graph G is going to contain *at most one* of the graphs B_i, i.e., G will either be $G_S \cup G_T$ (in the disconnected case), and other wise $G = G^{(i)} := (G_S \cup G_T \cup B_i)$ for *some* $i \in [k]$ in the connected case. As such, Claim 6.10.4 and the choice of d guarantees that G is a legitimate instance of the promise problem $\mathsf{ExpanderConn}_n$, i.e., that it is both sparse and has a constant spectral gap for each connected component as required.

We proceed with a standard adversarial argument. Without loss of generality, we assume that the query algorithm only queries edges $e \in \bigcup_{i=1}^{k} E(B_i)$. Whenever the query algorithm queries an edge e that belongs to B_i, the adversary declares that B_i, *as well as all B_j's for which $e \in B_j$* are *not* present in G. Claim 6.10.4 guarantees that at most $O(\log n)$ graphs B_js are excluded for each one query. Therefore, the adversary can continue with the aforementioned strategy for $t = \Omega(k/\log n) = \Omega(n/d \log n) = \Omega(n/\log n)$ steps, and still there will be at least one *unqueried* graph B_{i^*}. Therefore, if the query algorithm makes less than t queries to G, the adversary can either declare B_{i^*} is present or not (determining whether G is connected or not), contradicting the algorithm's output. □ Lemma 6.10.3

Theorem 6.7 now follows immediately from Lemma 6.10.3, Proposition 6.10.2, and Proposition 6.10.1.

Remark 6.10.5 (Extension to the PRAM model)**.** In Theorem 6.7 we proved the lower bound for $\mathsf{ExpanderConn}_n$ in the MPC model as our main focus in this chapter is on this model after all. However, our proof of Theorem 6.7 implies that $\mathsf{ExpanderConn}_n$ requires $\Omega(\log n)$ rounds in the (EREW) PRAM model as well. The reason is that our proof implies $\mathsf{ExpanderConn}_n$ is a *critical* function of $\Omega(n/\log n)$ variables: its output depends on the existence or non-existence of the $k = \Omega(n/\log n)$ expanders graphs $B_1, \ldots, B_k$. By results of [106, 265] computing such a function requires $\Omega(\log n)$ rounds in the (EREW) PRAM model.

Part II

Submodular Optimization on Massive Datasets

Chapter 7
Coverage Problems in the Streaming Model

Starting from this chapter, we switch to the second part of the thesis and focus on submodular optimization problems over massive datasets. We begin with our results for two canonical examples of submodular optimization problems, namely set cover and maximum coverage problems, collectively referred to as coverage problems, in the streaming setting. The materials in this chapter are based on [32, 27] with additional simplifications.

Introduced originally by Saha and Getoor [281], streaming coverage problems have been studied extensively in the streaming literature [281, 108, 40, 128, 116, 187, 174, 90, 50, 129, 247], resulting in numerous efficient algorithms for these problems. However, several fundamental questions regarding the complexity of these problems in the streaming setting have remained unresolved. In particular,

(i) *How well can we approximate the set cover problem in only a single-pass over the stream? (cf. [187] and its follow-up version in [174])*

A shortcoming of all previous streaming algorithms for the streaming set cover problem was that they either required strictly more than one pass over the stream, or achieved very large approximation ratio of $O(\sqrt{n})$ (where n is the number of elements in the universe). As such, obtaining *single-pass* streaming algorithms for set cover with sublinear space over the stream was a main open problem in this area.

(ii) *What is the space-approximation tradeoff for set cover and maximum coverage in multi-pass streams?*

While numerous upper bounds were known on the space-approximation tradeoff for coverage problems (see, e.g. [116, 187, 174, 90, 50, 247]), it was not clear what is the "right" space-approximation tradeoff for these problems due to lack of essentially any lower bounds.

We resolve both of these fascination questions in this chapter. In particular, we establish *tight* bounds on the space-approximation tradeoff for single-pass streaming algorithms for the set cover problem, hence settling the first question above. Our results rule out the existence of any non-trivial single-pass algorithm for the streaming set cover problem and hence provide a strong negative answer to the aforementioned open question. This also suggests a strong separation between power of single-pass vs multi-pass (even only *two*-pass) streaming algorithms for this problem. We further study the space-approximation tradeoff for multi-pass streaming algorithms for both set cover and coverage problems. We (slightly) improve the state-of-the-art algorithms for set cover that are allowed multiple passes over

the stream and more importantly also prove that the space-approximation tradeoff achieved by this algorithm is information-theoretically optimal, hence settling the second question above for set cover as well. Qualitatively similar results for the maximum coverage problem are also presented.

HighLights of Our Contributions

In this chapter, we will establish:

- Tight upper and lower bounds on the space complexity of single-pass streaming algorithms for approximating the set cover problem (Section 7.4).
- Tight upper and lower bounds on the space-approximation tradeoff of multi-pass streaming algorithms for set cover and maximum coverage (Sections 7.6 and 7.7).

7.1. Background

As we remarked in Section 2.4, coverage problems have been studied extensively in the literature. However, these results typically focused on the tradeoff between approximation guarantee and time complexity of the coverage problems. Nevertheless, in many settings, *space complexity* of the algorithms is crucial to optimize. A canonical example is in applications in big data analysis such as data mining and information retrieval [22, 281], web host analysis [99], operation research [164], and many others. In these settings, one would like to design algorithms capable of processing massive datasets using only few passes over the input and limited space. The well-established *streaming model* of computation [17, 254] precisely captures this setting.

In the *streaming set cover* problem, originally introduced by Saha and Getoor [281], the input sets are provided one by one in a stream and the algorithms are allowed to make a small number of passes over the stream while maintaining a sublinear space $o(mn)$ for processing the stream. The streaming set cover problem and the closely related maximum coverage problem have received quite a lot of attention in recent years [281, 108, 128, 116, 40, 187, 174, 90, 247, 50].

Particularly relevant to our work, Demaine *et al.* [116], have shown an α-approximation algorithm that uses $O(\alpha)$ passes over the stream and needs $\widetilde{O}(mn^{\Theta(1/\log \alpha)})$ space. Recently, Har-Peled *et al.* [174] provide a significant improvement over this algorithm: they developed an α-approximation, $O(\alpha)$-pass streaming algorithm that requires $\widetilde{O}(mn^{\Theta(1/\alpha)})$ space (see also [50]). For semi-streaming algorithm with much lower space, i.e., $\widetilde{O}(n)$ space, Emek and Rosen [128] presented an $O(\sqrt{n})$-approximation single-pass streaming algorithm using $O(n)$ space. This result was further generalized to allow multiple passes by Chakrabarti and Wirth [90] resulting in a p-pass streaming $O(n^{1/(p+1)})$-approximation algorithm with

$O(n)$ space for set cover; the pass-approximation tradeoff of this algorithm was also proved to be optimal by the same authors [90].

For maximum coverage, McGregor and Vu [247] and Bateni *et al.* [50] presented single-pass $(1+\varepsilon)$-approximation algorithms that use $\widetilde{O}(mk/\varepsilon^2)$ and $\widetilde{O}(m/\varepsilon^3)$ space, respectively. Badanidiyuru *et al.* [40] also presented a single-pass semi-streaming algorithm for this problem that achieves an almost 2-approximation using $O(n)$ space.

7.2. Our Results and Techniques

We now formally define α-approximation streaming algorithms for coverage problems. For brevity, we only formalize this notion for the set cover problem; the extension to maximum coverage is straightforward.

Definition 7.1 (α-approximation)**.** *An algorithm* ALG *is said to* α-approximate *the set cover problem iff on every input instance* $\mathcal{S}$*,* ALG *outputs a collection of (the indices of) at most* $\alpha \cdot \text{opt}$ *sets that covers* $[n]$*, along with a* certificate of covering *which, for each element* $e \in [n]$*, specifies the set used for covering* e*. If* ALG *is a randomized algorithm, we require that the certificate corresponds to a valid set cover w.p. at least* $2/3$*.*

The requirement of returning a certificate of covering in the above definition is standard in the literature (see, e.g., [128, 90, 174]).

7.2.1. Single-pass Streaming Complexity of Set Cover

We resolve the space complexity of the single-pass streaming set cover problem:

Result 7.1. *For any* $\alpha = o(\sqrt{n}/\log n)$ *and any* $m = \text{poly}(n)$*, there is a* deterministic *single-pass streaming algorithm that* α-approximates *the set cover problem using space* $\widetilde{O}(mn/\alpha)$ *bits. Moreover, any single-pass streaming algorithm (possibly* randomized*) that* α*-approximates the set cover problem must use* $\Omega(mn/\alpha)$ *bits of space.*

We should point out right away that in this chapter, we are *not* concerned with poly-time computability. However, in all hard instances we consider in this chapter, the minimum set cover size and the parameter k in maximum coverage) are *small constants* and hence these instances are trivially solvable in polynomial time in the classical (offline) setting. Our results hence establish the "hardness" of these instances under the space restrictions of the streaming model, independent of the NP-hardness of approximating these problems.

An α-approximation using $\widetilde{O}(mn/\alpha)$ bits of space can be simply achieved as follows. Merge (i.e., take the union of) every α sets in the stream into a single set; at the end of the stream, solve the set cover problem over the merged sets. To recover a certificate of covering, we also record for each element e in each merged set, any set in the merge that covers e.

It is an easy exercise to verify that this algorithm indeed achieves an α-approximation and can be implemented in space $\widetilde{O}(mn/\alpha)$ bits. Quite surprisingly, our Result 7.1 establishes that this simple algorithm is essentially the best possible; any α-approximation algorithm for the set cover problem requires $\widetilde{\Omega}(mn/\alpha)$ space.

Prior to our work, the best known lower bounds for single-pass streams ruled out $(3/2-\varepsilon)$-approximation using $o(mn)$ space [187] (see also [174]), $o(\sqrt{n})$-approximation in $o(m)$ space [128, 90], and $O(1)$-approximation in $o(mn)$ space [116] (only for *deterministic* algorithms). Note that these lower bound results leave open the possibility of a single-pass randomized $(3/2)$-approximation or even a deterministic $O(\log n)$-approximation algorithm for the set cover problem using only $\widetilde{O}(m)$ space. Our Result 7.1 on the other hand, rules out the possibility of having any *non-trivial* trade-off between the approximation factor and the space requirement, answering an open question raised by Indyk *et al.* [187] (see also [174]) in the strongest sense possible.

We should also point out that the bound of $\alpha = o(\sqrt{n}/\log n)$ in our lower bound is tight up to an $O(\log n)$ factor since an $O(\sqrt{n})$-approximation is known to be achievable in $\widetilde{O}(n)$ space (essentially independent of m for $m = \text{poly}(n)$) [128, 90].

We note that our lower bound in Result 7.1 crucially exploits the fact that the algorithm needs to output an actual set cover not only it size. In fact, in our paper [32], we show that this later task is strictly easier by providing an $\widetilde{O}(mn/\alpha^2)$ space algorithm for α-estimating the size of optimal set cover (without finding the actual sets) and prove that this bound is also optimal, using a simpler variant of our lower bound for α-approximation algorithms. For brevity, in this thesis, we only focus on the approximation algorithms for set cover and refer the interested reader to [32] for more details on estimation algorithms for set cover.

Our Techniques for Single-pass Streaming Set Cover

As is typical in the streaming literature, our lower bound is obtained by establishing *communication complexity* lower bounds; in particular, in the *one-way two-player* communication model (see Proposition 2.7.10). To prove this bound, we use the *information complexity* paradigm, which allows one to reduce the problem, via a direct sum type argument, to multiple instances of a simpler problem (see Sections 2.8 and 2.8.2 for a quick reminder of these notions). For this purpose, we introduce and analyze a new intermediate problem called the *Trap* problem

The Trap problem is a *non-boolean* problem defined as follows: Alice is given a set S, Bob is given a set E such that all elements of E belong to S except for a *special element* e^*, and the goal of the players is to "trap" this special element, i.e., to find a *small* subset of E which contains e^*. For our purpose, Bob only needs to trap e^* in a set of cardinality $|E|/2$. To prove a lower bound for the Trap problem, we design a novel reduction from the

well-known *Index* problem (again, see Section 2.7.2 for definition), which requires Alice and Bob to use the protocol for the Trap problem over non-legal inputs (i.e., the ones for which the Trap problem is not well-defined), while ensuring that they are not being "fooled" by the output of the Trap protocol over these inputs.

7.2.2. Multi-pass Streaming Complexity of Coverage Problems

Our main result in this part is a tight resolution of the space-approximation tradeoff for the streaming set cover problem:

Result 7.2. *Any streaming α-approximation* polylog(n)*-pass algorithm for the set cover problem requires $\widetilde{\Omega}(mn^{1/\alpha})$ space even on random arrival streams. This lower bound holds even for algorithms that are only required to output the value of optimal solution.*

Prior to our work, the best known lower bounds for *randomized multi-pass* streaming algorithms ruled out the possibility of $(\log n/2)$-approximation in p passes and $o(m/p)$ space [258], and exact solution in p passes and $o(mn^{1/2p})$ space [174] (the later holds only if $m = O(n)$). These results left open the possibility of obtaining, say, a 2-approximation in two passes or even an exact answer in $O(\log n)$ passes and $\widetilde{O}(m)$ space. On the other hand, Result 7.2 smoothly extends the bounds in [258] to the whole range of approximation factors $\alpha = o(\log n)$, proving the first *super-linear* in m lower bound for approximating set cover in *multi-pass* streams. It also significantly improves the bounds in [174] to $\widetilde{\Omega}(mn/p)$ (and all range of $m = \text{poly}(n)$) for p pass streaming algorithms that recover an exact answer[1].

As mentioned earlier, Har-Peled *et al.* [174] designed an α-approximation algorithm for the set cover problem that requires $\widetilde{O}(mn^{\Theta(1/\alpha)})$ space (for some unspecified constant larger than 2 in the Θ-notation in the exponent). We can show that with proper modifications, this algorithm in fact only requires $\widetilde{O}(mn^{1/\alpha})$ space (see Theorem 7.7), hence proving a *tight* upper bound for Result 7.2 (up to logarithmic factors), even for α-approximation algorithms (not only estimation). These results together resolve the space-approximation tradeoff for streaming set cover problem in multi-pass streams.

Finally, we point out that the lower bound in Result 7.2 is quite *robust* in the sense that it holds even when the sets are arriving in a random order. This is particularly relevant to the streaming set cover problem as most known techniques for this problem are based on element and set sampling and a-priori one may expect that random arrival streams can facilitate the use of such techniques, resulting in better bounds than the ones achievable in adversarial streams. We point that in general, many streaming problems are known to be distinctly easier in random arrival streams compared to adversarial streams (see,

[1]Note that this result also implies that the "right" tradeoff between space and number of passes for obtaining an *exact* solution to the streaming set cover is in fact linear as opposed to exponential, i.e., n/p as opposed to $n^{1/p}$, as was previously shown in [174].

e.g., [168, 221, 205] as well as our own results in Chapter 4 for maximum matching).

We further show an application of our techniques in establishing Result 7.2 to the *streaming maximum coverage* problem.

Result 7.3. *Any streaming* $(1+\varepsilon)$*-estimation (and so* $(1+\varepsilon)$*-approximation)* polylog(n)*-pass algorithm for the maximum coverage problem requires* $\widetilde{\Omega}(m/\varepsilon^2)$ *space even on random arrival streams. This lower bound applies even for the case* $k = O(1)$*.*

Single-pass $(1+\varepsilon)$-approximation algorithms for this problem that use, respectively, $\widetilde{O}(mk/\varepsilon^2)$ space and $\widetilde{O}(m/\varepsilon^3)$ have been proposed recently in [247, 50], and [50]. Our Result 7.3 is hence *tight* for any $k = O(1)$ (up to logarithmic factors) and within an $O(1/\varepsilon)$ factor of the best upper bound for the larger values of k.

McGregor and Vu [247] have very recently proved an $\widetilde{\Omega}(m)$ lower bound for polylog(n)-pass streaming algorithms that approximate the maximum coverage problem to within a factor better than $(\frac{e}{e-1})$ (a single-pass streaming algorithm with the same approximation guarantee in $\widetilde{O}(m)$ space is also developed in [247, 50]). The importance of Result 7.3 is thus in establishing the tight dependence on the parameter ε for this problem. This is important as $(1+\varepsilon)$-approximation algorithms for this problem for very small values of ε, i.e., $\varepsilon = 1/n^{\Omega(1)}$, are typically used as a sub-routine in approximating the streaming set cover problem in multiple passes [116, 174, 50] (see Section 7.6.4 for more details).

En route, we also obtain the following result which may be of independent interest: the communication complexity of computing an exact solution to the set cover problem or the maximum coverage problem in the two-player communication model is $\widetilde{\Omega}(mn)$ bits (see Theorems 7.6 and 7.10). This improves upon the previous $\Omega(m)$ lower bounds of Nisan [258] (for set cover) and McGregor and Vu [247] (for maximum coverage).

Techniques. Our techniques in establishing results in this part are also based on communication complexity tools and in particular information complexity and direct sum results (see Section 2.8.2). However, to prove multi-pass streaming lower bounds, we now need to work with two-way communication complexity (see Proposition 2.7.9) which is significantly more challenging. We elaborate on these challenges later in Section 7.5.

7.3. Preliminaries

Concentration Bounds. We use an extension of the Chernoff-Hoeffding bound for *negatively correlated* random variables. Random variables $\mathsf{X}_1, \ldots, \mathsf{X}_n$ are negatively correlated if for every set $S \subseteq [n]$, $\Pr(\wedge_{i \in S} \mathsf{X}_i = 1) \leq \prod_{i \in S} \Pr(\mathsf{X}_i = 1)$. It was first proved in [264] that the Chernoff-Hoeffding bound (for the upper tail) continues to hold for the case of random variables that satisfy this generalized version of negative correlation (see also [185]).

Proposition 7.3.1 ([264]). *Let $X_1, \ldots, X_n$ be negatively correlated random variables taking values in $[0,1]$ and let $X := \sum_{i=1}^{n} X_i$. Then, for any $0 < \varepsilon \leq 1$,*

$$\Pr\left(X \geq (1+\varepsilon)\,\mathbb{E}\left[X\right]\right) \leq \exp\left(-\frac{\varepsilon^2 \cdot \mathbb{E}\left[X\right]}{3}\right).$$

7.4. A Single-pass Lower bound for α-Approximate Set Cover

In this section, we prove that the simple α-approximation algorithm described in Section 7.1 is in fact optimal in terms of the space requirement. The following theorem formalizes the second part of Result 7.1.

Theorem 7.4. *For any $\alpha = o(\frac{\sqrt{n}}{\log n})$ and $m = \text{poly}(n)$, any* randomized *single-pass streaming algorithm that α-approximates the set cover problem with probability at least $2/3$ requires $\Omega(mn/\alpha)$ bits of space.*

Fix a (sufficiently large) value for n, $m = \text{poly}(n)$ (also $m = \Omega(\alpha \log n)$), and $\alpha = o(\frac{\sqrt{n}}{\log n})$; throughout this section, $\mathsf{SetCover}_{\mathsf{apx}}$ refers to the problem of α-approximating the set cover problem for instances with $m+1$ sets[2] defined over the universe $[n]$ in the one-way communication model, whereby the sets are partitioned between Alice and Bob.

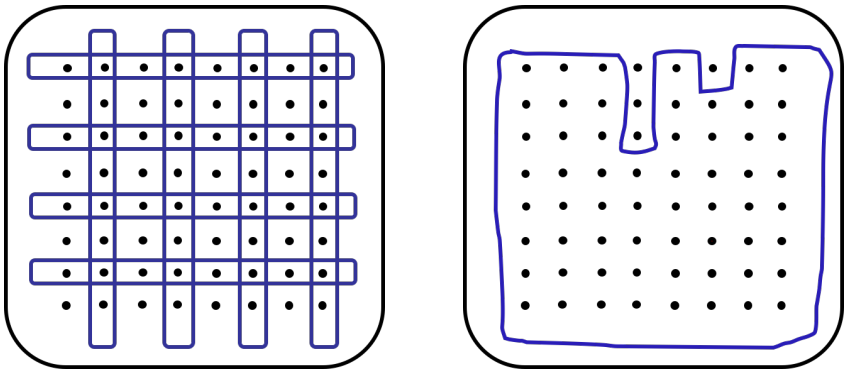
Overview. We design a hard input distribution $\mathcal{D}_{\mathsf{apx}}$ for $\mathsf{SetCover}_{\mathsf{apx}}$, whereby Alice is provided with a collection of m sets $S_1, \ldots, S_m$, each of size (roughly) n/α and Bob is given a *single* set T of size (roughly) $n - 2\alpha$. The input to the players are *correlated* such that there exists a set S_{i^*} in Alice's collection (i^* is unknown to Alice), such that $S_{i^*} \cup T$ covers all elements in $[n]$ except for a single *special element*. This in particular ensures that the optimal set cover size in this distribution is at most 3 w.h.p.

On the other hand, we "hide" this special element among the 2α elements in $\bar{T}$ in a way that if Bob does not have (essentially) full information about Alice's collection, he cannot even identify a set of α elements from $\bar{T}$ that contain this special element (w.p. strictly more than half). This implies that in order for Bob to be sure that he returns a valid set cover, he should additionally cover a majority of $\bar{T}$ with sets *other than* S_{i^*}. We design the distribution in a way that the sets in Alice's collection are "far" from each other and hence Bob is forced to use a *distinct* set for (roughly) each element in $\bar{T}$ that he needs to cover with sets other than S_{i^*}. This implies that Bob needs to output a set cover of size α (i.e., an $(\alpha/3)$-approximation) to ensure that every element in $[n]$ is covered.

A Hard Input Distribution for $\mathsf{SetCover}_{\mathsf{apx}}$

Consider the following distribution (see Figure 5).

[2] To simplify the exposition, we use $m+1$ instead of m as the number of sets.



(a) Alice's inputs is a collection of near-orthogonal sets.

(b) Bob's input is a single set containing all elements minus a small subset from a single set S_{i^*} of Alice plus one more element e^*.

Figure 5: Illustration of the hard distribution $\mathcal{D}_{\mathsf{apx}}$. Black dots are elements of the universe.

Distribution $\mathcal{D}_{\mathsf{apx}}$. A hard input distribution for $\mathsf{SetCover}_{\mathsf{apx}}$.

Notation. Let $\mathcal{F}$ be the collection of all subsets of $[n]$ with size $\frac{n}{10\alpha}$, and $\ell := 2\alpha \log m$.

- **Alice.** The input of Alice is a collection of m sets $\mathcal{S} = (S_1, \ldots, S_m)$, where for any $i \in [m]$, S_i is a set chosen independently and uniformly at random from $\mathcal{F}$.
- **Bob.** Pick an $i^* \in [m]$ (called the *special index*) uniformly at random; the input to Bob is a set $T = [n] \setminus E$, where E is chosen uniformly at random from all subsets of $[n]$ with $|E| = \ell$ and $|E \setminus S_{i^*}| = 1$.[a]

[a] Since $\alpha = o(\sqrt{n}/\log n)$ and $m = \text{poly}(n)$, the size of E is strictly smaller than the size of S_{i^*}.

The claims below summarize some useful properties of the distribution $\mathcal{D}_{\mathsf{apx}}$.

Claim 7.4.1. *For any instance $(\mathcal{S}, T) \sim \mathcal{D}_{\mathsf{apx}}$, w.p. $1 - o(1)$, $\text{opt}(\mathcal{S}, T) \leq 3$.*

Proof. Let e^* denote the element in $E \setminus S_{i^*}$. $\mathcal{S}^{-i^*}$ contains $m - 1$ random subsets of $[n]$ of size $n/10\alpha$, drawn independent of the choice of e^*. Therefore, each set in $\mathcal{S}^{-i^*}$ covers e^* with probability $1/10\alpha$. The probability that none of these $m - 1$ sets covers e^* is at most

$$(1 - 1/10\alpha)^{m-1} \leq (1 - 1/10\alpha)^{\Omega(\alpha \log n)} \leq \exp(-\Omega(\alpha \log n)/10\alpha). = o(1)$$

Hence, with probability $1 - o(1)$, there is at least one set $S \in \mathcal{S}^{-i^*}$ that covers e^*. Now, it is straightforward to verify that (S_{i^*}, T, S) form a valid set cover. □

Lemma 7.4.2. *With probability $1 - o(1)$, no collection of 3α sets from $\mathcal{S}^{-i^*}$ covers more*

than $\ell/2$ elements of E.

Proof. Recall that the sets in $\mathcal{S}^{-i^*}$ and the set E are chosen independent of each other. For each set $S \in \mathcal{S}^{-i^*}$ and for each element $e \in E$, we define an indicator binary random variable X_e, where $\mathsf{X}_e = 1$ iff $e \in S$. Let $\mathsf{X} := \sum_e \mathsf{X}_e$, which is the number of elements in E covered by S. We have,

$$\mathbb{E}[\mathsf{X}] = \sum_e \mathbb{E}[\mathsf{X}_e] = \frac{|E|}{10\alpha} = \frac{\log m}{5}.$$

Moreover, the variables X_e are negatively correlated since for any set $S' \subseteq E$,

$$\Pr\left(\bigwedge_{e \in S'} \mathsf{X}_e = 1\right) = \frac{\binom{n-|S'|}{\frac{n}{10\alpha}-|S'|}}{\binom{n}{\frac{n}{10\alpha}}} = \frac{\left(\frac{n}{10\alpha}\right) \cdot \left(\frac{n}{10\alpha}-1\right) \ldots \left(\frac{n}{10\alpha}-|S'|+1\right)}{(n) \cdot (n-1) \ldots (n-|S'|+1)}$$
$$\leq \left(\frac{1}{10\alpha}\right)^{|S'|} = \prod_{e \in S'} \Pr\left(\mathsf{X}_e = 1\right).$$

Hence, by the extended Chernoff bound (see Proposition 7.3.1), $\Pr\left(\mathsf{X} \geq \frac{\log m}{3}\right) = o(\frac{1}{m})$.

Therefore, using union bound over all $m-1$ sets in $\mathcal{S}^{-i^*}$, with probability $1-o(1)$, no set in $\mathcal{S}^{-i^*}$ covers more than $\log m/3$ elements in E, which implies that any collection of 3α sets can only cover up to $3\alpha \cdot \log m/3 = \ell/2$ elements in E. □

The Lower Bound for the Distribution $\mathcal{D}_{\mathsf{apx}}$

In order to prove our lower bound for $\mathsf{SetCover}_{\mathsf{apx}}$ on $\mathcal{D}_{\mathsf{apx}}$, we define an intermediate communication problem which we call the *Trap* problem.

Problem 7.4.3 (*Trap* problem)**.** *Alice is given a set $S \subseteq [n]$ and Bob is given a set $E \subseteq [n]$ such that $E \setminus S = \{e^*\}$; Bob needs to output a set $L \subseteq E$ with $|L| \leq |E|/2$ such that $e^* \in L$.*

In the following, we use Trap to refer to the trap problem with $|S| = n/10\alpha$ and $|E| = \ell = 2\alpha \log m$ (notice the similarity to the parameters in $\mathcal{D}_{\mathsf{apx}}$). We define the following distribution $\mathcal{D}_{\mathsf{Trap}}$ for Trap. Alice is given a set $S \in_R \mathcal{F}$ (recall that $\mathcal{F}$ is the collection of all subsets of $[n]$ of size $n/10\alpha$) and Bob is given a set E chosen uniformly at random from all sets that satisfy $|E \setminus S| = 1$ and $|E| = 2\alpha \log m$. We first use a direct sum style argument to prove that under the distributions $\mathcal{D}_{\mathsf{apx}}$ and $\mathcal{D}_{\mathsf{Trap}}$, information complexity of solving $\mathsf{SetCover}_{\mathsf{apx}}$ is essentially equivalent to solving m copies of Trap. Formally,

Lemma 7.4.4. *For any constant $\delta < 1/2$, any δ-error protocol π_{SC} on $\mathcal{D}_{\mathsf{apx}}$ can be used to obtain a $(\delta + o(1))$-error protocol π_{Trap} on $\mathcal{D}_{\mathsf{Trap}}$ such that $\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) \leq \frac{1}{m} \cdot \mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{apx}}}(\pi_{SC})$.*

Proof. The protocol π_{Trap} is as follows.

Protocol π_{Trap}. The protocol for solving Trap using a protocol π_{SC} for $\mathsf{SetCover}_{\mathsf{apx}}$.

Input: An instance $(S, E) \sim \mathcal{D}_{\mathsf{Trap}}$.

Output: A set L with $|L| \leq |E|/2$, such that $e^* \in L$.

---

1. Using *public randomness*, the players sample an index $i^* \in [m]$ uniformly at random.
2. Alice creates a tuple $\mathcal{S} = (S_1, \ldots, S_m)$ by assigning $S_{i^*} = S$ and sampling each remaining set uniformly at random from $\mathcal{F}$ using *private randomness*. Bob creates a set $T := \bar{E}$.
3. The players run the protocol π_{SC} over the input $(\mathcal{S}, T)$.
4. Bob computes the set L of all elements in $E = \bar{T}$ whose certificate (i.e., the set used to cover them) is not S_{i^*}, and outputs L.

We first argue the correctness of π_{Trap} and then bound its information cost. To argue the correctness, notice that the distribution of instances of $\mathsf{SetCover}_{\mathsf{apx}}$ constructed in the reduction is exactly $\mathcal{D}_{\mathsf{apx}}$. Consequently, it follows from Claim 7.4.1 that, with probability $1 - o(1)$, any α-approximate set cover can have at most 3α sets. Let $\widehat{\mathcal{S}}$ be the set cover computed by Bob minus the sets S_{i^*} and T. As $e^* \in E = \bar{T}$ and moreover is *not* in S_{i^*}, it follows that e^* should be covered by some set in $\widehat{\mathcal{S}}$. This means that the set L that is output by Bob contains e^*. Moreover, by Lemma 7.4.2, the number of elements in E covered by the sets in $\widehat{\mathcal{S}}$ is at most $\ell/2$ w.p. $1 - o(1)$. Hence, $|L| \leq \ell/2 = |E|/2$. This implies that:

$$\Pr_{\mathcal{D}_{\mathsf{Trap}}}\Big(\pi_{\mathsf{Trap}} \text{ errs}\Big) \leq \Pr_{\mathcal{D}_{\mathsf{apx}}}\Big(\pi_{\mathsf{SC}} \text{ errs}\Big) + o(1) \leq \delta + o(1).$$

We now bound the information cost of π_{Trap}. Let I be the random variable for the choice of $i^* \in [m]$ in the protocol π_{Trap} (which is uniform in $[m]$). By Proposition 2.8.4 on information cost of one-way protocols, we have,

$$\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) = \mathop{\mathbb{E}}_{i \sim \mathsf{I}}\Big[\mathbb{I}(\Pi_{\mathsf{Trap}}\,; \mathsf{S} \mid \mathsf{I} = i)\Big] = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathsf{S}_i \mid \mathsf{I} = i) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathsf{S}_i),$$

where the last two equalities hold since (i) the joint distribution of Π_{SC} and S_i conditioned on $\mathsf{I} = i$ under $\mathcal{D}_{\mathsf{Trap}}$ is equivalent to the one under $\mathcal{D}_{\mathsf{apx}}$, and (ii) the random variables Π_{SC} and S_i are jointly independent of the event $\mathsf{I} = i$ (by the definition of $\mathcal{D}_{\mathsf{apx}}$) and hence we can "drop" the conditioning on this event. We can further derive,

$$\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) = \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathsf{S}_i) \leq \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathsf{S}_i \mid \mathsf{S}^{<i}).$$

The inequality holds since S_i and $\mathsf{S}^{<i}$ are independent and conditioning on independent variables can only increase the mutual information (by Proposition 2.6.3). Finally,

$$\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) \leq \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathsf{S}_i \mid \mathsf{S}^{<i}) \underset{\text{Fact 2.6.1-(6)}}{=} \frac{1}{m} \cdot I_{\mathcal{D}_{\mathsf{apx}}}\Big(\Pi_{\mathsf{SC}}; \mathsf{S}\Big) = \frac{1}{m} \cdot \mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{apx}}}(\pi_{\mathsf{SC}}),$$

where the first equality is by the chain rule for mutual information. □

Having established Lemma 7.4.4, our task now is to lower bound the information complexity of Trap over the distribution $\mathcal{D}_{\mathsf{Trap}}$. We prove this lower bound using a novel reduction from the well-known *Index* problem (see Section 2.7.2), denoted by Index^n_k. In Index^n_k over the distribution $\mathcal{D}_{\mathsf{Index}}$, Alice is given a set $A \subseteq [n]$ of size k chosen uniformly at random and Bob is given an element a such that w.p. $1/2$ $a \in_R A$ and w.p. $1/2$ $a \in_R [n] \setminus A$; Bob needs to determine whether $a \in A$ (the Yes case) or not (the No case).

We remark that similar distributions for Index^n_k have been previously studied in the literature (see, e.g., [280], Section 3.3). The proof of the following lemma is standard and hence we omit it here (see, e.g., [32] for this proof).

Lemma 7.4.5. *For any $k < n/2$ and constant $\delta' < 1/2$, any δ'-error protocol π_{Index} for Index^n_k on $\mathcal{D}$ has (external) information cost $\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Index}}}(\pi_{\mathsf{Index}}) = \Omega(k)$.*

Using Lemma 7.4.5, we prove the following lemma, which is the key part of the proof.

Lemma 7.4.6. *For any constant $\delta < 1/2$, any δ-error protocol π_{Trap} for Trap on $\mathcal{D}_{\mathsf{Trap}}$ has (external) information cost $\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) = \Omega(n/\alpha)$.*

Proof. Let $k = n/10\alpha$; we design a δ'-error protocol π_{Index} for Index^n_k using any δ-error protocol π_{Trap} (over $\mathcal{D}_{\mathsf{Trap}}$) as a subroutine, for some constant $\delta' < 1/2$.

Protocol π_{Index}. The protocol for reducing Index^n_k to Trap.

Input: An instance $(A, a) \sim \mathcal{D}_{\mathsf{Index}}$.

Output: Yes if $a \in A$ and No otherwise.

---

1. Alice picks a set $B \subseteq A$ with $|B| = \ell - 1$ uniformly at random using *private randomness*.
2. To invoke the protocol π_{Trap}, Alice creates a set $S := A$ and sends the message $\pi_{\mathsf{Trap}}(S)$, along with the set B to Bob.
3. If $a \in B$, Bob outputs Yes and terminates the protocol.
4. Otherwise, Bob constructs a set $E = B \cup \{a\}$ and computes $L := \pi_{\mathsf{Trap}}(S, E)$ using

the message received from Alice.

5. If $a \in L$, Bob outputs No, and otherwise outputs Yes.

We should note right away that the distribution of instances for Trap defined in the previous reduction does *not* match $\mathcal{D}_{\mathsf{Trap}}$. Therefore, we need a more careful argument to establish the correctness of the reduction.

We prove this lemma in two claims; the first claim establishes the correctness of the reduction and the second one proves an upper bound on the information cost of π_{Index} based on the information cost of π_{Trap}.

Claim 7.4.7. *π_{Index} is a δ'-error protocol for Index_k^n over $\mathcal{D}_{\mathsf{Index}}$ for the parameter $k = n/10\alpha$ and a constant $\delta' < 1/2$.*

Proof. Let R denote the *private* coins used by Alice to construct the set B. Also, define $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}}$ (resp. $\mathcal{D}^{\mathsf{N}}_{\mathsf{Index}}$) as the distribution of Yes instances (resp. No instances) of $\mathcal{D}_{\mathsf{Index}}$. We have,

$$\Pr_{\mathcal{D}_{\mathsf{Index}},\mathsf{R}}\Big(\pi_{\mathsf{Index}} \text{ errs}\Big) = \frac{1}{2} \cdot \Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}\Big(\pi_{\mathsf{Index}} \text{ errs}\Big) + \frac{1}{2} \cdot \Pr_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Index}},\mathsf{R}}\Big(\pi_{\mathsf{Index}} \text{ errs}\Big). \tag{7.1}$$

Note that we do not consider the randomness of the protocol π_{Trap} (used in construction of π_{Index}) as it is independent of the randomness of the distribution $\mathcal{D}_{\mathsf{Index}}$ and the private coins R. We now bound each term in Eq (7.1) separately. We first start with the easier case which is the second term.

The distribution of instances (S, E) for Trap created in the reduction by the choice of $(A, a) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Index}}$ and the randomness of R, is the same as the distribution $\mathcal{D}_{\mathsf{Trap}}$. Moreover, in this case, the output of π_{Index} would be wrong iff $a \in E \setminus S$ (corresponding to the element e^* in Trap) does not belong to the set L output by π_{Trap}. Hence,

$$\Pr_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Index}},\mathsf{R}}\Big(\pi_{\mathsf{Index}} \text{ errs}\Big) = \Pr_{\mathcal{D}_{\mathsf{Trap}}}\Big(\pi_{\mathsf{Trap}} \text{ errs}\Big) \leq \delta. \tag{7.2}$$

We now bound the first term in Equation (7.1). Note that when $(A, a) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}}$, there is a small chance that π_{Index} is "lucky" and a belongs to the set B (see Line (3) of the protocol). Let this event be $\mathcal{E}$. Conditioned on $\mathcal{E}$, Bob outputs the correct answer with probability 1; however note that probability of $\mathcal{E}$ happening is only $o(1)$. Now suppose $\mathcal{E}$ does not happen. In this case, the distribution of instances (S, E) created by the choice of $(A, a) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}}$ (and randomness of R) does *not* match the distribution $\mathcal{D}_{\mathsf{Trap}}$. However, we have the following important property:

Given that (S, E) is the instance of Trap created by choosing (A, a) from $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}}$ and sampling $\ell - 1$ random elements of A (using R), the element a is *uniform* over the set E.

In other words, knowing (S, E) does not reveal any information about the element a.

Note that since (S, E) is not chosen according to the distribution $\mathcal{D}_{\mathsf{Trap}}$ (actually it is not even a "legal" input for Trap), it is possible that π_{Trap} terminates, outputs a non-valid set, or outputs a set $L \subseteq E$. Unless $L \subseteq E$ (and satisfies the cardinality constraint), Bob is always able to determine that π_{Trap} is not functioning correctly and hence outputs Yes (and errs with probability at most $\delta < 1/2$). However, if $L \subseteq E$, Bob would not know whether the input to π_{Trap} is legal or not. In the following, we explicitly analyze this case.

In this case, L is a subset of E chosen by the (inner) randomness of π_{Trap} for a fixed S and E and moreover $|L| \leq |E|/2$ (by definition of Trap). The probability that π_{Index} errs in this case is exactly equal to the probability that $a \in L$. However, as stated before, for a fixed (S, E), the choice of L is independent of the choice of a and moreover, a is uniform over E; hence $a \in L$ happens with probability at most $1/2$. Formally, (here, $\mathsf{R}^{\mathsf{Trap}}$ denotes the inner randomness of π_{Trap})

$$\begin{aligned}
\Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}(\pi_{\mathsf{Index}} \text{ errs} \mid \bar{\mathcal{E}}) &= \Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}\left(a \in L = \pi_{\mathsf{Trap}}(\mathsf{S},\mathsf{E}) \mid \bar{\mathcal{E}}\right) \\
&= \mathop{\mathbb{E}}_{(\mathsf{S},\mathsf{E})|\bar{\mathcal{E}},\mathsf{R}^{\mathsf{Trap}}}\left[\Pr\left(a \in L \mid \mathsf{S} = S, \mathsf{E} = E, \mathsf{R}^{\mathsf{Trap}} = R^{\mathsf{Trap}}, \bar{\mathcal{E}}\right)\right] \\
&\qquad (L = \pi_{\mathsf{Trap}}(S, E) \text{ is a fixed set conditioned on } (S, E, R^{\mathsf{Trap}})) \\
&= \mathop{\mathbb{E}}_{(\mathsf{S},\mathsf{E})|\bar{\mathcal{E}}} \mathop{\mathbb{E}}_{\mathsf{R}^{\mathsf{Trap}}}\left[\frac{|L|}{|E|}\right],
\end{aligned}$$

since a is uniform on E conditioned on $(S, E, R^{\mathsf{Trap}})$ and $\bar{\mathcal{E}}$. Hence, $\Pr(\pi_{\mathsf{Index}} \text{ errs} \mid \bar{\mathcal{E}}) \leq \frac{1}{2}$, since by definition, for any output set L, $|L| \leq |E|/2$.

As stated earlier, whenever $\mathcal{E}$ happens, π_{Index} makes no error; hence,

$$\Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}(\pi_{\mathsf{Index}} \text{ errs}) = \Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}(\bar{\mathcal{E}}) \cdot \Pr_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Index}},\mathsf{R}}(\pi_{\mathsf{Index}} \text{ errs} \mid \bar{\mathcal{E}}) \leq \frac{1 - o(1)}{2}. \tag{7.3}$$

Finally, by plugging the bounds in Equations (7.2,7.3) in Equation (7.1) and assuming δ is bounded away from $1/2$, we have,

$$\Pr_{\mathcal{D}_{\mathsf{Index}},\mathsf{R}}(\pi_{\mathsf{Index}} \text{ errs}) \leq \frac{1}{2} \cdot \frac{1 - o(1)}{2} + \frac{1}{2} \cdot \delta = \frac{1 - o(1)}{4} + \frac{\delta}{2} \leq \frac{1}{2} - \varepsilon,$$

for some constant ε bounded away from 0. □ Claim 7.4.7

We now bound the information cost of π_{Index} under $\mathcal{D}_{\mathsf{Index}}$.

Claim 7.4.8. $\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Index}}}(\pi_{\mathsf{Index}}) \leq \mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) + O(\ell \log n)$.

Proof. We have,

$$\begin{aligned}
\mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Index}}}(\pi_{\mathsf{Index}}) &= \mathbb{I}(\Pi_{\mathsf{Index}}(\mathsf{A}) \,;\mathsf{A}) \\
&= \mathbb{I}(\Pi_{\mathsf{Trap}}(\mathsf{S}), \mathsf{B} \,;\mathsf{A}) \\
&= \mathbb{I}(\Pi_{\mathsf{Trap}}(\mathsf{S}) \,;\mathsf{A}) + \mathbb{I}(\mathsf{B} \,;\mathsf{A} \mid \Pi_{\mathsf{Trap}}(\mathsf{S})) \\
&\qquad\qquad \text{(by chain rule of mutual information, Fact 2.6.1-(6))} \\
&\leq \mathbb{I}(\Pi_{\mathsf{Trap}}(\mathsf{S}) \,;\mathsf{A}) + \mathbb{H}(\mathsf{B} \mid \Pi_{\mathsf{Trap}}(\mathsf{S})) \\
&\leq \mathbb{I}(\Pi_{\mathsf{Trap}}(\mathsf{S}) \,;\mathsf{A}) + O(\ell \log n) \qquad (|\mathsf{B}| = O(\ell \log n) \text{ and Fact 2.6.1-(1)}) \\
&= \mathbb{I}(\Pi_{\mathsf{Trap}}(\mathsf{S}) \,;\mathsf{S}) + O(\ell \log n) \qquad (\mathsf{A} = \mathsf{S} \text{ as defined in } \pi_{\mathsf{Index}}) \\
&= \mathsf{IC}^{\mathrm{ext}}_{\mathcal{D}_{\mathsf{Trap}}}(\pi_{\mathsf{Trap}}) + O(\ell \log n). \\
&\text{(the joint distribution of } (\Pi_{\mathsf{Trap}}(\mathsf{S}), \mathsf{S}) \text{ is identical under } \mathcal{D}_{\mathsf{Index}} \text{ and } \mathcal{D}_{\mathsf{Trap}})
\end{aligned}$$

□ Claim 7.4.8

The lower bound now follows from Claims 7.4.7 and 7.4.8, and Lemma 7.4.5 for the parameters $k = |S| = \frac{n}{10\alpha}$ and $\delta' < 1/2$, and using the fact that $\alpha = o(\sqrt{n}/\log n)$, $\ell = 2\alpha \log m$, and $m = \mathrm{poly}(n)$, and hence $\Omega(n/\alpha) = \omega(\ell \log n)$. □ Lemma 7.4.6

To conclude, by Lemma 7.4.4 and Lemma 7.4.6, for any set of parameters $\delta < 1/2$, $\alpha = o(\frac{\sqrt{n}}{\log n})$, and $m = \mathrm{poly}(n)$, external information complexity of $\mathsf{SetCover}_{\mathsf{apx}}$ on $\mathcal{D}_{\mathsf{apx}}$ is $\Omega(mn/\alpha)$. Since the information complexity is a lower bound on the communication complexity (Proposition 2.8.3), we have the following theorem.

Theorem 7.5. *For any constant $\delta < 1/2$, $\alpha = o(\frac{\sqrt{n}}{\log n})$, and $m = \mathrm{poly}(n)$, the one-way communication complexity of $\mathsf{SetCover}_{\mathsf{apx}}$ is:*

$$\mathsf{R}^{\text{1-way}}_{\delta}(\mathsf{SetCover}_{\mathsf{apx}}) = \Omega(mn/\alpha).$$

Finally, since one-way communication complexity is also a lower bound on the space complexity of single-pass streaming algorithms (see Proposition 2.7.10), we obtain Theorem 7.4 as a corollary of Theorem 7.5.

7.5. Technical Overview of Multi-pass Streaming Lower Bounds

We now switch to the second part of our results in this section on multi-pass lower bounds. In this section, we focus on providing a technical overview of the proof of Result 7.2 - Re-

sult 7.3 is also proven along similar lines. The starting point of our work is our Result 7.1 in which we proved a tight space lower bound for single-pass streaming algorithms of set cover by analyzing the *one-way communication complexity* of this problem. We extend our approach to lower bound the two-way communication complexity of the set cover problem and ultimately obtain the desired lower bound in Result 7.2 for multi-pass streaming algorithms. To do this, we need to address the following issues:

First, the type of distribution used in Result 7.1 is clearly not suitable for proving lower bounds in the two-way model. In particular, we need a distribution with both Alice and Bob having $\Omega(m)$ sets and additionally, no clear "signal" to either party as which of the sets are more important, i.e., correspond to the sets S_{i^*} and T in the above distribution. To achieve this, we first design a collection of sets $Z_1, \ldots, Z_m$ such that no collection of α sets Z_i's can cover the universe $[n]$ unless they contain a single set Z_{i^*} which is in fact equal to $[n]$ already (for remaining sets Z_i, we have $|Z_i| \approx n - n^{1-1/\alpha}$). Next, we decompose each Z_i into two sets S_i and T_i and provide Alice with S_i, and Bob with T_i. This way, the sets S_{i^*} and T_{i^*} form a set cover of size two and no other collection of α pairs (S_i, T_i) can cover the universe; we further prove that "mix and matching" the sets (i.e., picking S_i but not T_i or vice versa) in the solution is not helpful either, hence implying that any α-approximation algorithm for set cover needs to find the sets S_{i^*} and T_{i^*}.

The next step is to prove the lower bound for the above distribution. Unlike the lower bound in the one-way model that was based on hiding the content of the set S_{i^*}, here we need to argue that in fact the index i^* itself is hidden from the players (as otherwise, one more round of communication can reveal the content of the sets S_{i^*} and T_{i^*} as well). We again use the information complexity paradigm to prove the communication lower bound for this distribution. We embed different instances of the well-known *set disjointness* problem (see Section 2.7.2) in each pair (S_i, T_i) such that all embedded instances are *intersecting* except for the instance for S_{i^*} and T_{i^*} which is *disjoint*. As we seek a direct-sum style argument for two-way protocols, we need a more careful argument than our previous results that were tailored for one-way protocols. In particular, we now use the notion of *internal* information complexity (as opposed to *external* information complexity used in the last part; recall the distinction between these two measures described in Section 2.8) that allows us to use the powerful techniques developed in [46, 66, 74] to obtain the direct-sum result.

Finally, we need to lower bound the information complexity of the set disjointness problem on the specific distribution induced by the set cover instances. The set cover distribution is designed in a way to ensure that the distribution of underlying set disjointness instances matches the known hard input distributions for this problem. However, there is a subtlety here; known information complexity lower bounds for set disjointness (that we are aware of) are all over distributions that are supported only on disjoint sets, i.e., Yes-

instances of the problem (see, e.g., [44, 68, 298])[3]. However, for our purpose, we need to lower bound the information cost of set disjointness protocols on distributions that are intersecting. We achieve this using an application of the "information odometer" of [76] (and subsequent work in [162]) to relate the information cost of the protocols on Yes and No instances of the problem together and obtain the result.

We are not done though, as we seek a lower bound for random arrival streams and for this, we we follow the approach of [86] in proving the communication complexity lower bounds when the input data is *randomly allocated* between the players (as opposed to adversarial partitions). However, the distributions and the problem considered in this paper are different from the ones in [86].

7.6. Space-Approximation Tradeoff for Multi-Pass Set Cover

We prove our main result on the space-approximation tradeoff for the streaming set cover problem in this section. Formally,

Theorem 7.6. *For any $\alpha = o(\log n/\log\log n)$, $m = \text{poly}(n)$, and $p \geq 1$, any* randomized *algorithm that can make p passes over any collection of m subsets of $[n]$ presented in a random order stream and outputs an α-estimation to the optimal value of the set cover problem w.p. larger than $3/4$ (over the randomness of both the stream order and the algorithm) must use $\widetilde{\Omega}(mn^{\frac{1}{\alpha}}/p)$ space.*

Theorem 7.6 formalizes Result 7.2. We further prove that the tradeoff achieved in Theorem 7.6 is in fact tight up to logarithmic factors; this is achieved by performing some proper modifications to the algorithm of [174]. Formally,

Theorem 7.7. *There exists a streaming algorithm that for any integer $\alpha \geq 1$, and any parameter $\varepsilon > 0$, with high probability, computes an $(\alpha+\varepsilon)$-approximation to the streaming set cover problem using $(2\alpha+1)$ passes over the stream in adversarial order and $\widetilde{O}(mn^{1/\alpha}/\varepsilon^2 + n/\varepsilon)$ space.*

We emphasize that the main contribution of this section is in proving Theorem 7.6; we mainly present Theorem 7.7 to prove a matching upper bound on the bounds in Theorem 7.6, hence establishing a tight space-approximation tradeoff for the streaming set cover problem.

The rest of this section is mainly devoted to the proof of Theorem 7.6. We start by introducing some notation. In Section 7.6.1, we introduce a hard input distribution for the set cover problem in adversarial streams. We prove a lower bound for this distribution in Section 7.6.2. We extend this lower bound to random arrival streams in Section 7.6.3 and

[3] We remark that this is not just a coincidence and in fact is crucial for performing the typical reduction to the AND problem used in proving the lower bound for set disjointness, see, e.g., [298] for more details.

finish the proof of Theorem 7.6. Section 7.6.4 contains the proof of Theorem 7.7.

Notation. To prove Theorem 7.6, we prove a lower bound on the communication complexity of the set cover problem: Fix a (sufficiently large) value for n, $m = \text{poly}(n)$, and $\alpha = o(\log n / \log\log n)$; in this section, SetCover refers to the problem of α-estimating the optimal value of the set cover problem with $2m$ sets[4] defined over the universe $[n]$ in the two-player communication model, whereby the sets are partitioned between Alice and Bob.

7.6.1. A Hard Input Distribution for SetCover

We shall use the well-known *set-disjointness* communication problem (denoted by Disj) in proving Theorem 7.6. Fix an integer $t \geq 1$; in Disj_t, Alice and Bob are given two sets $A \subseteq [t]$ and $B \subseteq [t]$, and their goal is to return Yes if $A \cap B = \emptyset$ and No otherwise.

The following is a known hard distribution for Disj_t.

Distribution $\mathcal{D}_{\mathsf{Disj}}$. A hard input distribution for Disj_t.

- Start with $A = B = [t]$.
- For each element $e \in [t]$ independently: w.p. $1/3$ drop e from both A and B, w.p. $1/3$ drop e from A, and w.p. $1/3$ drop e from B.
- Pick $Z \in_R \{0,1\}$ uniformly at random. If $Z = 1$, pick a uniformly at random element $e^* \in [t]$ and let A and B both contain e^* (if $Z = 0$, keep the sets as before).

We further use $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$ and $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ to denote, respectively, the distribution of Yes and No instances of Disj on $\mathcal{D}_{\mathsf{Disj}}$; in other words, $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}} := (\mathcal{D}_{\mathsf{Disj}} \mid Z = 0)$ and $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}} := (\mathcal{D}_{\mathsf{Disj}} \mid Z = 1)$.

The following proposition on the (internal) information complexity of Disj is well-known.

Proposition 7.6.1 (cf. [44, 68])**.** *For any $\delta < 1/2$ and any δ-error protocol π_{Disj} of Disj_t on the distribution $\mathcal{D}_{\mathsf{Disj}}$, the internal information cost of π_{Disj} is: $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \Omega(t)$.*

Let t be an integer to be determined later; we use the distribution $\mathcal{D}_{\mathsf{Disj}}$ for Disj_t to design a hard input distribution for SetCover. Before that, we need a couple of definition.

Definition 7.2 (Mapping-extension)**.** *For the two sets $[t]$ and $[n]$, we define a* mapping-extension *of $[t]$ to $[n]$ as a function $f : [t] \mapsto 2^{[n]}$, whereby for each $i \in [t]$, $f(i) \subseteq [n]$ is mapped to n/t* unique *elements in $[n]$. Similarly, for any set $A \subseteq [t]$, we abuse the notation and define $f(A) := \bigcup_{i \in A} f(i)$.*

We are now ready to define our hard input distribution for SetCover.

[4] To simplify the exposition, we use $2m$ instead of m as the number of sets.

Distribution $\mathcal{D}_{\mathsf{SC}}$. A hard input distribution for SetCover.

Notation. Let $t := 2^{-15} \cdot \left(\frac{n}{\log m}\right)^{\frac{1}{\alpha}}$ and $\mathcal{F}$ be the set of all mapping-extensions of $[t]$ to $[n]$.

- For each $i \in [m]$:
  - Let $(A_i, B_i) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ for Disj_t and pick $f_i \in_R \mathcal{F}$ uniformly at random.
  - Let $S_i = [n] \setminus f_i(A_i)$ and $T_i := [n] \setminus f_i(B_i)$.
- Pick $\theta \in_R \{0, 1\}$ uniformly at random. If $\theta = 0$, do nothing, otherwise:
  - Sample $i^* \in_R [m]$ uniformly at random.
  - Resample $(A_{i^*}, B_{i^*}) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$ for Disj_t and redefine S_{i^*} and T_{i^*} as before using the new pair (A_{i^*}, B_{i^*}).
- Let the input to Alice and Bob be $\mathcal{S} := \{S_i\}_{i \in [m]}$ and $\mathcal{T} := \{T_i\}_{i \in [m]}$, respectively.

In the following, we use Z to denote any set in $\mathcal{S} \cup \mathcal{T}$, i.e., when it is not relevant whether it belongs to $\mathcal{S}$ or $\mathcal{T}$. For a collection of sets $\mathcal{Z} = \{Z_1, \ldots, Z_\ell\}$, we use $C(\mathcal{Z})$ to denote the set of elements that $\mathcal{Z}$ covers, i.e., $C(\mathcal{Z}) := \bigcup_{i=1}^{\ell} Z_i$. We say that $\mathcal{Z}$ is a *singleton-collection*, if for any $i \in [m]$, at least one of S_i or T_i is *not* present in $\mathcal{Z}$. In contrast, we say that $\mathcal{Z}$ is a *pair-collection*, if for all $i \in [m]$, $S_i \in \mathcal{Z}$ iff $T_i \in \mathcal{Z}$ as well.

Remark 7.6.2. *A few remarks are in order:*

1. *W.h.p., for any $i \in [m]$, $|S_i| = 2n/3 \pm o(n)$ and $|T_i| = 2n/3 \pm o(n)$.*
   (Proof. follows from the definition of the distribution $\mathcal{D}_{\mathsf{Disj}}$ and Chernoff bound).
2. *For any $i \in [m]$, conditioned on $|S_i| = \ell$, the set S_i is chosen uniformly at random from all ℓ-subsets of $[n]$; similarly for T_i*
3. *For any $i \in [m]$, $S_i \cup T_i = [n] \setminus f_i(A_i \cap B_i)$. Moreover, whenever $(A_i, B_i) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$, the set $f_i(A_i \cap B_i)$ is a (n/t)-subset of $[n]$ chosen uniformly at random.*
   (Proof. the first part follows from the fact that f_i maps each $j \in [t]$ to unique elements; the second part is by the random choice of $f_i \in_R \mathcal{F}$ and the fact that $|A_i \cap B_i| = 1$).
4. *Whenever $\theta = 0$, for any $i \neq j$, the sets $Z_i \in \{S_i, T_i\}$ and $Z_j \in \{S_j, T_j\}$ are chosen independent of each other ($Z_i \perp Z_j$).*

Let $\mathrm{opt}(\mathcal{S}, \mathcal{T})$ denote the size of an optimal set cover in the instance $(\mathcal{S}, \mathcal{T})$. It follows from Remark 7.6.2-(3) that whenever $\theta = 1$ in the distribution $\mathcal{D}_{\mathsf{SC}}$, $\mathrm{opt}(\mathcal{S}, \mathcal{T}) = 2$; simply take S_{i^*} and T_{i^*} and since $A_{i^*} \cap B_{i^*} = \emptyset$, they cover the whole universe. In the following, we prove that when $\theta = 0$, $\mathrm{opt}(\mathcal{S}, \mathcal{T})$ is relatively large. This implies that any α-approximation protocol for SetCover has to essentially determine the value of θ. In the next section, we

prove that this task requires a large communication by the players.

Lemma 7.6.3. *For* $(\mathcal{S},\mathcal{T}) \sim \mathcal{D}_{\mathsf{SC}}$*:* $\Pr\left(\mathrm{opt}(\mathcal{S},\mathcal{T}) > 2\alpha \mid \theta = 0\right) = 1 - o(1)$.

For the proof of Lemma 7.6.3, we need the following auxiliary lemma that upper bounds the number of elements that a collection of large random sets can cover. The proof is standard and is omitted here (but can be found in our paper [27])

Lemma 7.6.4 ([27])**.** *Let* $\mathcal{S} = \{S_1, \ldots, S_k\}$ *be a collection of* $(n-s)$*-subsets of* $[n]$ *that are chosen independently and uniformly at random. Suppose* $U \subseteq [n]$ *is another set chosen independent of* $\mathcal{S}$*; if* $k = o(e^s)$*, then,*

$$\Pr\left(|U \setminus (S_1 \cup \ldots \cup S_k)| < \frac{|U|}{2} \cdot \left(\frac{s}{2n}\right)^k\right) < 2 \cdot \exp\left(-\frac{|U|}{8} \cdot \left(\frac{s}{2n}\right)^k\right).$$

Proof of Lemma 7.6.3. Let $\mathcal{C}$ be any collection of 2α sets from $(\mathcal{S},\mathcal{T})$. We bound the probability that $\mathcal{C}$ covers the universe $[n]$ entirely, i.e., is a feasible set cover, and then use a union bound on all possible choices for $\mathcal{C}$ to finalize the proof. In the following, we condition on the event $\mathcal{E}_1$ that states that $|S_i| \leq 3n/4$ and $|T_i| \leq 3n/4$ for all $i \in [m]$ (which happens with probability $1 - o(1)$ by Remark 7.6.2-(1)).

Partition the collection $\mathcal{C}$ into a pair-collection $\mathcal{C}_P$, and a singleton-collection $\mathcal{C}_S$ (this partitioning is always possible and unique by definition). We first lower bound the number of elements that are not covered by the singleton-collection:

Claim 7.6.5. $\Pr\left(\left|\overline{C(\mathcal{C}_S)}\right| \leq \frac{n}{2^{6\alpha+1}} \mid \mathcal{E}_1\right) \leq 1 - \frac{1}{m^{\omega(\alpha)}}$.

Proof. Let $\mathcal{C}_S := \{Z_1, \ldots, Z_k\}$; clearly $k = |\mathcal{C}_S| \leq |\mathcal{C}| = 2\alpha$. Without loss of generality, we assume that $k = 2\alpha$. By conditioning on the event $\mathcal{E}_1$ and Remark 7.6.2-(2), we know that each Z_i is an ℓ_i-subset of $[n]$, for some $\ell_i \leq 3n/4$, chosen uniformly at random from all ℓ_i-subsets of $[n]$. Again without loss of generality, we simply increase the size of each Z_i so that they all have size exactly $3n/4$. Moreover, since no two sets S_i and T_i are both simultaneously present in $\mathcal{C}_S$, by Remark 7.6.2-(4), all sets in $\mathcal{C}_S$ are chosen independently.

Consequently, by Lemma 7.6.4, for $U = [n]$, $s = n/4$, and collection $\mathcal{C}_S$, we have,

$$\Pr\left(\left|\overline{C(\mathcal{C}_S)}\right| < \frac{n}{2} \cdot \left(\frac{1}{8}\right)^{2\alpha} \mid \mathcal{E}_1\right) < 2 \cdot \exp\left(-\frac{n}{8} \cdot \left(\frac{1}{8}\right)^{2\alpha}\right).$$

A simplification of the above equation, plus using the fact that $\alpha = o(\log n / \log\log n)$, and hence $n/2^{\Theta(\alpha)} = \omega(\alpha \log m)$, proves the final result. □ Claim 7.6.5

Let $\mathcal{E}_2$ be the event that $\left|\overline{C(\mathcal{C}_S)}\right| \geq \frac{n}{2^{6\alpha+1}}$; in the following, we condition on this event.

Now consider the sets in the pair-collection $\mathcal{C}_P$. For any pair $(S_i, T_i) \in \mathcal{C}_P$, we define $C_i := S_i \cup T_i$. Note that there are at most α different possible sets C_i. By Remark 7.6.2-(3), the sets C_i's are random sets of size $(n - n/t)$, and by Remark 7.6.2-(4), they are chosen independent of each other. By Lemma 7.6.4, for $U = \overline{C(\mathcal{C}_S)}$, $s = n/t$, and collection of sets C_i's, we have, $\Pr\left(U \setminus (C(\mathcal{C}_P)) = \emptyset \mid \mathcal{E}_1, \mathcal{E}_2\right) \leq 2 \cdot \exp\left(-\frac{n}{2^{6\alpha+4}} \cdot \left(\frac{1}{2t}\right)^{\alpha}\right) \leq \frac{1}{m^{3\alpha}}$. We conclude,

$$\begin{aligned}
\Pr\left(\text{opt}(\mathcal{S}, \mathcal{T}) \leq 2\alpha\right) &\leq \Pr\left(\bar{\mathcal{E}}_1\right) + \Pr\left(\exists\ \mathcal{C} \text{ that covers } [n] \mid \mathcal{E}_1\right) \\
&\leq \Pr\left(\bar{\mathcal{E}}_1\right) + \sum_{\mathcal{C}} \left(\Pr\left(\bar{\mathcal{E}}_2 \mid \mathcal{E}_1\right) + \Pr\left(C(\mathcal{C}) = [n] \mid \mathcal{E}_1, \mathcal{E}_2\right)\right) \\
&\leq o(1) + \binom{m}{2\alpha} \cdot \left(\frac{1}{m^{\omega(\alpha)}} + \frac{1}{m^{3\alpha}}\right) = o(1)
\end{aligned}$$

proving the lemma. □ Lemma 7.6.3

7.6.2. The Lower Bound for the Distribution $\mathcal{D}_{\mathsf{SC}}$

Throughout this section, fix π_{SC} as a δ-error protocol for SetCover on the distribution $\mathcal{D}_{\mathsf{SC}}$. We first show that protocol π_{SC} is essentially solving m copies of the Disj_t problem on the distribution $\mathcal{D}_{\mathsf{Disj}}$ (for the parameter t in the distribution $\mathcal{D}_{\mathsf{SC}}$) and then use a direct-sum style argument (similar in spirit to the ones in [46, 66, 74] and Proposition 2.8.5) to argue that the information cost of π_{SC} shall be m times larger than the information complexity of solving Disj_t. However, to make the direct-sum argument work, we can only consider π_{SC} on the distribution $\mathcal{D}_{\mathsf{SC}} \mid \theta = 0$, i.e., when *all* underlying Disj_t instances are sampled from $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$. Consequently, we can only lower bound the information cost of π_{SC} based on the information complexity of Disj_t on the distribution $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$.

Lemma 7.6.6. *There exists a $(\delta + o(1))$-protocol π_{Disj} for Disj_t on the distribution $\mathcal{D}_{\mathsf{Disj}}$ such that:*

1. $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \frac{O(1)}{m} \cdot \mathsf{IC}^{\text{int}}_{\mathcal{D}_{\mathsf{SC}}}(\pi_{\mathsf{SC}})$.
2. $\|\pi_{\mathsf{Disj}}\| = \|\pi_{\mathsf{SC}}\|$.

Proof. We design the protocol π_{Disj} as follows:

Protocol π_{Disj}. The protocol for solving Disj_t using a protocol π_{SC} for SetCover.

Input: An instance $(A, B) \sim \mathcal{D}_{\mathsf{Disj}}$. **Output:** Yes if $A \cap B = \emptyset$ and No otherwise.

1. Using public randomness, the players sample an index $i^* \in_R [m]$ and m mapping-extensions $f_1, \ldots, f_m$ independently and uniformly at random from $\mathcal{F}$.
2. Using public randomness, the players sample the sets $A^{<i^*}$ and $B^{>i^*}$ each from $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$

independently.

3. Using private randomness, Alice samples the sets $A^{>i^*}$ such that $(A_j, B_j) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ (for all $j > i^*$); similarly Bob samples the sets $B^{<i^*}$.
4. The players construct the collections $\mathcal{S} := \{S_1, \ldots, S_m\}$ and $\mathcal{T} := \{T_1, \ldots, T_m\}$ by setting $S_i := [n] \setminus f_i(A_i)$ and $T_i := [n] \setminus f_i(B_i)$ (exactly as in distribution $\mathcal{D}_{\mathsf{SC}}$).
5. The players solve the SetCover instance using π_{SC} and output No iff π_{SC} estimates $\text{opt}(\mathcal{S}, \mathcal{T}) \leq 2\alpha$ and Yes otherwise.

It is easy to see that the distribution of instances $(\mathcal{S}, \mathcal{T})$ created in the protocol π_{Disj} matches the distribution $\mathcal{D}_{\mathsf{SC}}$ for SetCover exactly. Moreover, by Lemma 7.6.3, $\text{opt}(\mathcal{S}, \mathcal{T}) > 2\alpha$ w.p. $1 - o(1)$, whenever $(A, B) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ and $\text{opt}(\mathcal{S}, \mathcal{T}) = 2$ whenever $(A, B) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$. Consequently, since π_{SC} is an α-approximation protocol,

$$\Pr\left(\pi_{\mathsf{Disj}} \text{ errs}\right) \leq \Pr\left(\pi_{\mathsf{SC}} \text{ errs}\right) + o(1) \leq \delta + o(1),$$

and hence π_{Disj} is indeed a $(\delta + o(1))$-error protocol for Disj on the distribution $\mathcal{D}_{\mathsf{Disj}}$. Moreover, it is clear that the communication cost of π_{Disj} is at most the communication cost of π_{SC}. We now prove the bound on the information cost of this protocol.

Our goal is to bound the information cost of π_{Disj} whenever the instance (A, B) is sampled from $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$. Let F be a random variable denoting the tuple $(f_1, \ldots, f_m)$, I be a random variable for i^* and R be the set of public randomness. By Proposition 2.8.2,

$$\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{A} \mid \mathsf{B}, \mathsf{R}) + \mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{B} \mid \mathsf{A}, \mathsf{R}).$$

We now bound the first term in the RHS above (the second term is bounded the same).

$$\begin{aligned}
\mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{A} \mid \mathsf{B}, \mathsf{R}) &= \mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{A} \mid \mathsf{B}, \mathsf{R}, \mathsf{I}) && (\mathsf{I} \text{ is chosen using public randomness}) \\
&= \mathop{\mathbb{E}}_{i \sim \mathsf{I}} \left[\mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{A}_i \mid \mathsf{B}_i, \mathsf{A}^{<i}, \mathsf{B}^{>i}, \mathsf{F}, \mathsf{I} = i)\right] && (\mathsf{R} = (\mathsf{A}^{<i}, \mathsf{B}^{>i}, \mathsf{F}, \mathsf{I})) \\
&= \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{Disj}}\, ; \mathsf{A}_i \mid \mathsf{B}_i, \mathsf{A}^{<i}, \mathsf{B}^{>i}, \mathsf{F}),
\end{aligned}$$

where the last equality is true since conditioned on $(A, B) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$, all sets A_j, B_j (for $j \in [m]$) are chosen from $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ and hence are independent of the even "$I = i$"[5].

[5] We point out that this is the exact reason we need to consider information cost of π_{Disj} on $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ (instead of $\mathcal{D}_{\mathsf{Disj}}$) as otherwise (A_j, B_j)'s are *not* independent of $I = i$ and hence this equality would not hold.

Define $\boldsymbol{A} := (\mathsf{A}_1, \ldots, \mathsf{A}_m)$ and $\boldsymbol{B} := (\mathsf{B}_1, \ldots, \mathsf{B}_m)$; we can further derive,

$$\begin{aligned}
\mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathsf{A} \mid \mathsf{B}, \mathsf{R}) &= \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathsf{A}_i \mid \mathsf{B}_i, \mathsf{A}^{<i}, \mathsf{B}^{>i}, \mathsf{F}) \leq \frac{1}{m} \cdot \sum_{i=1}^{m} \mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathsf{A}_i \mid \mathsf{A}^{<i}, \boldsymbol{B}, \mathsf{F}) \\
&\qquad (\mathsf{A}_i \perp \mathsf{B}^{<i} \mid \boldsymbol{B}, \mathsf{F} \text{ and hence we can apply Proposition 2.6.3}) \\
&= \frac{1}{m} \cdot \mathbb{I}(\Pi_{\mathsf{Disj}}\,; \boldsymbol{A} \mid \boldsymbol{B}, \mathsf{F}) = \frac{1}{m} \cdot \mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathcal{S} \mid \mathcal{T}, \mathsf{F}) \\
&\qquad (\text{chain rule of mutual information, Fact 2.6.1-(6)}) \\
&= \frac{1}{m} \cdot \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}, F, \theta = 0)
\end{aligned}$$

where the second last equality is because $\boldsymbol{A}$ (resp. $\boldsymbol{B}$) and $\mathcal{S}$ (resp. $\mathcal{T}$) determine each other conditioned on F, and last equality is because the distribution of set cover instances and the messages communicated by the players under $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ and under $\mathcal{D}_{\mathsf{SC}} \mid \theta = 0$ exactly matches. Moreover,

$$\begin{aligned}
\mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathsf{A} \mid \mathsf{B}, \mathsf{R}) &\leq \frac{1}{m} \cdot \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}, \mathsf{F}, \theta = 0) \leq \frac{2}{m} \cdot \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}, \mathsf{F}, \theta) \\
&\qquad (\text{by definition of mutual information as } \Pr(\theta = 0) = 1/2) \\
&\leq \frac{2}{m} \cdot (\mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}, \mathsf{F}) + \mathcal{H}(\theta)) = \frac{2}{m} \cdot \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}, \mathsf{F}) + \frac{2}{m} \\
&\qquad (\text{by Proposition 2.6.5 and Fact 2.6.1-(1)}) \\
&\leq \frac{2}{m} \cdot \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}) + \frac{2}{m} \\
&\qquad (\Pi_{\mathsf{SC}} \perp \mathsf{F} \mid \mathcal{S}, \mathcal{T} \text{ and hence we can apply Proposition 2.6.4})
\end{aligned}$$

By performing the same exact calculation for $\mathbb{I}(\Pi_{\mathsf{Disj}}\,; \mathsf{B} \mid \mathsf{A}, \mathsf{R})$, we obtain that,

$$\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) \leq \frac{2}{m} \cdot (\mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{S} \mid \mathcal{T}) + \mathbb{I}(\Pi_{\mathsf{SC}}\,; \mathcal{T} \mid \mathcal{S})) + \frac{4}{m} = \frac{O(1)}{m} \cdot \mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{SC}}}(\pi_{\mathsf{SC}})$$

where in the last inequality we used the fact that information cost of π_{SC} is at least 1. This finalizes the proof of the lemma. □

Recall that in Lemma 7.6.6, we bound the information cost of π_{Disj} on the distribution $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ (as opposed to $\mathcal{D}_{\mathsf{Disj}}$); in the following we prove that this weaker bound is still sufficient for our purpose.

Lemma 7.6.7. *For any $\delta < 1/2$, any δ-error protocol π_{Disj} for Disj_t on $\mathcal{D}_{\mathsf{Disj}}$ with $\|\pi_{\mathsf{Disj}}\| = 2^{o(t)}$ has internal information cost $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{N}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \Omega(t)$.*

By Proposition 7.6.1, any δ-error protocol for Disj_t on $\mathcal{D}_{\mathsf{Disj}}$ has $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \Omega(t)$ (notice again that the information cost is measured on the distribution $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$ (and not $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ needed

in Lemma 7.6.7). From this, it is also easy to obtain that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \Omega(t)$. However, to prove Lemma 7.6.7, we need to lower bound the information cost of π_{Disj} under $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$.

To achieve this, we can relate the information costs $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}})$ and $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}})$ to each other. The goal is to argue that if there is a large discrepancy in the information cost of π_{Disj} on $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$ and $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$, then the information cost of the protocol itself can be used to distinguish between these two cases. We can achieve this goal using an elegant construction of an "information odometer" by [76]; informally speaking, the odometer allows the players to "keep track" of the amount of information revealed in a protocol (i.e., the information cost of the protocol), while incurring a relatively small additional information cost overhead.

Intuitively, we can use the odometer to argue that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \Theta(\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}))$ as follows: suppose towards a contradiction that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = \tau$ for some $\tau = o(\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}))$ and consider a new protocol π'_{Disj} for Disj on $\mathcal{D}_{\mathsf{Disj}}$ which runs π_{Disj} and the information odometer for π_{Disj} in parallel. Whenever the odometer estimates the information cost of π_{Disj} to be larger than $c \cdot \tau$ (for some sufficiently large constant c), the players terminate the protocol and declare that the answer for Disj is No (as information cost of π_{Disj} on $\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}$ is typically not much more than τ, while its information cost on $\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}$ is $\omega(\tau)$). If the cost is not estimated more than $c \cdot \tau$ by the end of the protocol, the players output the same answer as in π_{Disj}. As the information cost of the information odometer itself is bounded by $O(\tau)$, this results in protocol π'_{Disj} to have $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) = o(t)$, a contradiction. This argument was first made explicit in [162].

Lemma 7.6.8 ([162])**.** *Fix any function F, constants $0 < \varepsilon_1 < \varepsilon_2 < 1/2$, input distribution $\mathcal{D}$, and define $\mathcal{D}^{\mathsf{N}} := \mathcal{D} \mid F^{-1}(\mathsf{No})$. For every ε_1-error protocol π for F on $\mathcal{D}$, there exists an ε_2-error protocol π' for F on $\mathcal{D}$ such that: $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}}(\pi') = O\left(\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}}(\pi) + \log \|\pi\|\right)$.*

Equipped with this lemma, we can now prove Lemma 7.6.7 easily.

Proof of Lemma 7.6.7. Let π'_{Disj} be any δ'-error protocol for Disj on $\mathcal{D}_{\mathsf{Disj}}$ for $\delta' < 1/2$. We first prove that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) = \Omega(t)$ using the fact that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) = \Omega(t)$ as follows:

$$\begin{aligned}
\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) &= \mathbb{I}(\Pi'_{\mathsf{Disj}}\,; \mathsf{A} \mid \mathsf{B}) + \mathbb{I}(\Pi'_{\mathsf{Disj}}\,; \mathsf{B} \mid \mathsf{A}) \\
&\geq \mathbb{I}(\Pi'_{\mathsf{Disj}}\,; \mathsf{A} \mid \mathsf{B}, \theta) + \mathbb{I}(\Pi'_{\mathsf{Disj}}\,; \mathsf{B} \mid \mathsf{A}, \theta) - 2\mathcal{H}(\theta) && \text{(by Proposition 2.6.5)} \\
&= \frac{1}{2} \cdot \mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) - 2 && (\text{as } \mathcal{D}^{\mathsf{Y}}_{\mathsf{Disj}} = \mathcal{D}_{\mathsf{Disj}} \mid \theta = 0) \\
&= \Omega(t). && \text{(by Proposition 7.6.1)}
\end{aligned}$$

Now suppose towards a contradiction that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}})$ is $o(t)$. We can then apply

Lemma 7.6.8 for the function $F = \mathsf{Disj}$, $\varepsilon_1 = \delta$ and $\varepsilon_2 = \delta' < 1/2$ to obtain a protocol π'_{Disj} with $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{Disj}}}(\pi'_{\mathsf{Disj}}) = O\left(\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}}(\pi_{\mathsf{Disj}}) + \log \|\pi_{\mathsf{Disj}}\|\right)$ which is $o(t)$; a contradiction. □

Theorem 7.8. *For any constant $\delta < 1/2$, $\alpha = o(\frac{\log n}{\log\log n})$, and $m = \mathrm{poly}(n)$, the two-way communication complexity of* SetCover *is* $\mathsf{R}_\delta(\mathsf{SetCover}) = \widetilde{\Omega}(mn^{\frac{1}{\alpha}})$.

Proof. Let $t = \Theta\left((\frac{n}{\log m})^{\frac{1}{\alpha}}\right)$ and suppose towards a contradiction that there exists a δ-error protocol π_{SC} for SetCover on the distribution $\mathcal{D}_{\mathsf{SC}}$ with $\|\pi_{\mathsf{SC}}\| = o(mt)$; by Proposition 2.8.3, $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}_{\mathsf{SC}}}(\pi_{\mathsf{SC}}) = o(mt)$ also. By Lemma 7.6.6, this implies that there exists a $(\delta + o(1))$-error protocol π_{Disj} for Disj on the distribution $\mathcal{D}_{\mathsf{Disj}}$ such that $\mathsf{IC}^{\mathrm{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{Disj}}}(\pi_{\mathsf{Disj}}) = o(t)$, and $\|\pi_{\mathsf{Disj}}\| = o(mt) \leq 2^{o(t)}$ (since $m = \mathrm{poly}(n)$ and $\alpha = o(\frac{\log n}{\log\log n})$). However, this is in contradiction with Lemma 7.6.7, implying that $\|\pi_{\mathsf{SC}}\| = \Omega(mt)$, hence proving the theorem. □

As a corollary of Theorem 7.8 (combined with Proposition 2.7.9), we have that the space complexity of any α-approximation streaming algorithm for set cover that uses $\mathrm{polylog}(n)$ passes on *adversarial streams* is $\widetilde{\Omega}(mn^{\frac{1}{\alpha}})$. In the next section, we extend this result to random arrival streams and complete the proof of Theorem 7.6.

7.6.3. Proof of Theorem 7.6

The distribution $\mathcal{D}_{\mathsf{SC}}$ used in the previous section is quite "adversarial" and as such is not suitable for proving the lower bound for random arrival streams. In order to prove the lower bound in Theorem 7.6 for random arrival streams, we need to relax the adversarial partitioning of the sets in the distribution $\mathcal{D}_{\mathsf{SC}}$ to a randomized partition.

Distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$. A random partitioning of the distribution $\mathcal{D}_{\mathsf{SC}}$

- Sample the collections $(\mathcal{S}, \mathcal{T}) \sim \mathcal{D}_{\mathsf{SC}}$.
- Assign each set in $\mathcal{S} \cup \mathcal{T}$ to Alice w.p. $1/2$ and the remainings to Bob.

We show that even this seemingly easier distribution still captures all the "hardness" of distribution $\mathcal{D}_{\mathsf{SC}}$. Formally,

Lemma 7.6.9. *For any constant $\delta < 1/4$, $\alpha = o(\frac{\log n}{\log\log n})$, and $m = \mathrm{poly}(n)$, the distributional communication complexity of* SetCover *is* $\mathsf{D}_{\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}},\delta}(\mathsf{SetCover}) = \widetilde{\Omega}(mn^{\frac{1}{\alpha}})$.

Proof. Let $\mathcal{S} = \{S_1, \ldots, S_m\}$ and $\mathcal{T} = \{T_1, \ldots, T_m\}$ be the collections of sets sampled from $\mathcal{D}_{\mathsf{SC}}$ in the distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$. For a sampled instance in $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$, we say that the index $i \in [m]$ is *good* iff S_i is given to one player and T_i to another. Let $G \subseteq [m]$ be the collection of all good indices. The index i^* is chosen independent of the random partitioning in $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$, and hence the probability that $i^* \in G$ is exactly $|G|/m$. Let $\mathcal{E}$ denote the event that

$|G| \geq m/2 - o(m)$ and $i \in G$. We have,

$$\begin{aligned}\Pr(\mathcal{E}) &= \Pr(|G| \geq m/2 - o(m)) \cdot \Pr(i^* \in G \mid |G| \geq m/2 - o(m)) \\ &\geq \Pr(|G| \geq (1 - o(1)) \cdot \mathbb{E}[G]) \cdot \frac{1 - o(1)}{2} \geq (1 - o(1)) \cdot \frac{1}{2},\end{aligned}$$

where the last inequality is by Chernoff bound. Now fix a δ-error protocol π_{SC} for $\mathsf{SetCover}$ on the distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$. Then,

$$\Pr(\pi_{\mathsf{SC}} \text{ errs} \mid \mathcal{E}) \leq \frac{\Pr(\pi_{\mathsf{SC}} \text{ errs})}{\Pr(\mathcal{E})} \leq 2\delta + o(1) \tag{7.4}$$

This in particular implies that there exists a set $G^\star \subseteq [n]$ with $|G^\star| \geq m/2 - o(m)$, such that conditioned on the set of good indices being $G^\star$ and conditioned on $i^* \in G^\star$, the probability that π_{SC} errs is at most $2\delta + o(1)$. Note that conditioned on the aforementioned events, the index i^* is chosen from $G^\star$ uniformly at random. This implies that the distribution of the input given to Alice and Bob limited to the sets in $G^\star$ matches the distribution $\mathcal{D}_{\mathsf{SC}}$ (with the number of the sets being $2 \cdot |G^\star|$ instead of $2m$). We can then use this to embed an instance of $\mathsf{SetCover}$ over the distribution $\mathcal{D}_{\mathsf{SC}}$ into the sets $G^\star$ and obtain a protocol π'_{SC} for $\mathcal{D}_{\mathsf{SC}}$.

More formally, the protocol π'_{SC} works as follows: Given an instance $(\mathcal{S}', \mathcal{T}')$ sampled from $\mathcal{D}_{\mathsf{SC}}$ (with $|\mathcal{S}'| = |\mathcal{T}'| = |G^\star|$), Alice and Bob use public coins to complete their input (i.e., increase the number of the sets to $2m$) by sampling from the distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$ conditioned on $G^\star$ (this is possible without any communication as the sets outside $G^\star$ are sampled independent of the sets in $G^\star$). The players then run the protocol π_{SC} on this new instance and return the same answer as this protocol. As the distribution of the $\mathsf{SetCover}$ instances sampled in the protocol π'_{SC} matches the distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$ conditioned on $G^\star$ and $i^* \in G^\star$, by Eq (7.4), the probability that π'_{SC} errs is at most $2\delta + o(1)$. Since $\delta < 1/4$, we obtain a δ'-error protocol for $\mathsf{SetCover}$ on the distribution $\mathcal{D}_{\mathsf{SC}}$ with $2|G^\star| = \Theta(m)$ sets and universe of size n, for a constant $\delta' < 1/2$. Consequently, by Theorem 7.8, $\|\pi_{\mathsf{SC}}\| = \|\pi'_{\mathsf{SC}}\| = \widetilde{\Omega}(|G^\star| \cdot n^{\frac{1}{\alpha}}) = \widetilde{\Omega}(mn^{\frac{1}{\alpha}})$, proving the lemma. □

Proof of Theorem 7.6. Fix a p-pass s-space streaming algorithm ALG for the set cover problem over random arrival streams that outputs an α-approximation w.p. at least $1 - \delta$ for $\delta < 1/4$. One can easily turn ALG into a δ-error protocol for $\mathsf{SetCover}$ on the distribution $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$: Alice and Bob take a random permutation of their inputs and then treat their combined input as a set stream and run ALG on that. The random partitioning of the input plus the random permutation taken by the players ensure that the constructed stream is a random permutation of the input sets. Consequently, this protocol is a δ-error protocol for

$\mathsf{SetCover}$ on $\mathcal{D}^{\mathsf{rnd}}_{\mathsf{SC}}$ that uses $O(p \cdot s)$ bits of communication. Since $\delta < 1/4$, by Lemma 7.6.9, $p \cdot s = \widetilde{\Omega}(mn^{\frac{1}{\alpha}})$, proving the theorem. □

7.6.4. An α-Approximation Algorithm for Streaming Set Cover

In this section, we prove the optimality of the lower bound in Theorem 7.6 by establishing a matching upper bound (i.e. Theorem 7.7). As stated earlier, our algorithm is a simple modification of the algorithm of [174]. In particular, we obtain our improved algorithm by using a one-shot pruning step as opposed to the iterative pruning of [174], and employing a more careful element sampling (compare the bounds in Lemma 7.6.13 in this paper with Lemma 2.5 in [174]).

In the following, we assume that we are given a value $\widetilde{\mathrm{opt}}$ which is a $(1+\varepsilon)$-approximation of opt, i.e., the optimal solution size of the given instance. This is without loss of generality as we can run the algorithm in parallel for $O(\log n/\varepsilon)$ guesses for $\widetilde{\mathrm{opt}} \in [1, n]$ and return the smallest computed set cover among all parallel runs.

The general idea behind the algorithm is as follows: we know that $\widetilde{\mathrm{opt}}$ sets are enough to cover the whole universe $[n]$; hence, if we find a $(1-\rho)$-approximate k-cover of the input sets for the parameter $k = \widetilde{\mathrm{opt}}$ and $\rho = 1/n^{1/\alpha}$, we can reduce the number of uncovered elements by a factor of $n^{1/\alpha}$. Repeating this process α times then results in a collection of at most $\alpha \cdot \widetilde{\mathrm{opt}}$ sets that covers the whole universe, i.e., an α-approximate set cover. It is worth mentioning that this is the general principle behind most (but not all) streaming algorithms for set cover, see, e.g. [174, 50, 281, 116].

Notice that we can readily use the maximum coverage streaming algorithms of [247, 50] as a sub-routine to find the approximate k-cover above; however, doing so would result in a sub-optimal algorithm for set cover as these algorithms have space dependency of (at least) $\Omega(m/\rho^2) = \Omega(mn^{2/\alpha})$ (even ignoring the dependence on k, i.e., $\widetilde{\mathrm{opt}}$). In fact, as we prove in the next section (see Result 7.3), any $(1-\rho)$-approximate k-cover algorithm needs $\Omega(m/\rho^2)$ space in general. To bypass this, we crucially use the fact that the aforementioned maximum coverage instances have the additional property that the optimal answer is the whole universe and hence the element sampling technique of [174] (and similar ones in [247, 50]) can be improved for this special case. We now provide the algorithm.

Algorithm 1. An α-approximation algorithm for the streaming set cover problem.

Input. A stream $\mathcal{S} = (S_1, \ldots, S_m)$ of subsets of $[n]$, and a $(1+\varepsilon)$-approximation $\widetilde{\mathrm{opt}}$ of $\mathrm{opt}(\mathcal{S})$.
Output. A collection of $(1+\varepsilon) \cdot \alpha \cdot \widetilde{\mathrm{opt}}$ sets that cover the universe.

1. Let $U \leftarrow [n]$ and $\mathsf{SOL} \leftarrow \emptyset$.
2. Make a single pass over the stream and if $|S_i \cap U| \geq n/(\varepsilon \cdot \widetilde{\text{opt}})$, then:
   (a) $\mathsf{SOL} \leftarrow \mathsf{SOL} \cup \{i\}$ and $U \leftarrow U \setminus S_i$.
3. For $j = 1$ to α iterations:
   (a) Let U_{smpl} be a subset of U chosen by picking each element independently and w.p. $p = 16 \cdot \widetilde{\text{opt}} \cdot \log m / n^{1-1/\alpha}$.
   (b) Make a single pass over the stream and for all $i \in [m]$, store $S'_i = S_i \cap U_{\mathsf{smpl}}$ in the memory.
   (c) Find an optimal set cover OPT' of the instance $(S'_1, \ldots, S'_m)$ and let $\mathsf{SOL} \leftarrow \mathsf{SOL} \cup \text{OPT}'$.
   (d) Make another pass over the stream and let $U_{\mathsf{smpl}} \leftarrow U_{\mathsf{smpl}} \setminus \bigcup_{i \in \text{OPT}'} S_i$.
4. Return SOL as a set cover of the input instance.

We start by bounding the space requirement of Algorithm 1 .

Lemma 7.6.10. *Algorithm 1 requires $\widetilde{O}(mn^{1/\alpha}/\varepsilon + n)$ space w.p. at least $1 - 1/m^2$.*

Proof. It is easy to see that maintaining SOL and U requires, respectively, $O(m)$ and $O(n)$ space. In the following, we analyze the space required for storing the sets $(S'_1, \ldots, S'_m)$. After the first pass of the algorithm, no set contains more than $n/(\varepsilon \cdot \widetilde{\text{opt}})$ elements in U. Fix a set $S_i \in \mathcal{S}$; we have,

$$\mathbb{E}\,|S_i \cap U_{\mathsf{smpl}}| = |S_i| \cdot p \leq n/(\varepsilon \cdot \widetilde{\text{opt}}) \cdot \left(16 \cdot \widetilde{\text{opt}} \cdot \log m / n^{1-1/\alpha}\right) = 16 \cdot n^{1/\alpha} \cdot \log m/\varepsilon$$

Hence, by Chernoff bound, w.p. $1 - 1/m^3$, $|S_i \cap U_{\mathsf{smpl}}| = \widetilde{O}(n^{1/\alpha}/\varepsilon)$. The final bound now follows from this and a union bound on all m sets in $\mathcal{S}$. □

The following two lemmas establish the correctness of the algorithm.

Lemma 7.6.11. *Algorithm 1 picks at most $(\alpha + \varepsilon) \cdot \widetilde{\text{opt}}$ sets in SOL.*

Proof. It is immediate to see that in the first pass, the algorithm picks at most $\varepsilon \cdot \widetilde{\text{opt}}$ sets as otherwise U would be empty. Moreover, in each subsequent α iterations, the algorithm picks at most $\widetilde{\text{opt}}$ sets since $(S'_1, \ldots, S'_m)$ has a set cover of size at most $\widetilde{\text{opt}}$ (as the original instance had a set cover of size $\leq \widetilde{\text{opt}}$). □

Lemma 7.6.12. *The set SOL in Algorithm 1 is a feasible set cover of $[n]$ w.p. $1 - 1/m$.*

To prove Lemma 7.6.12, we use the following property of element sampling that first appeared in [116] (similar ideas also appear in [247, 174]); for completeness we provide a

self-contained proof of this lemma here.

Lemma 7.6.13. *Let $0 < \rho < 1$ be a parameter and $\mathcal{S} = (S_1, \ldots, S_m)$ be a collection of m subsets of $[n]$ with $\mathrm{opt}(\mathcal{S}) \leq k$. Suppose U_{smpl} is a subset of $[n]$ obtained by picking each element independently and w.p. $p \geq 16 \cdot k \cdot \log m/(\rho \cdot n)$; then, w.p. $1 - 1/m^2$, any collection of k sets in $\mathcal{S}$ that covers U_{smpl} entirely also covers at least $(1-\rho) \cdot n$ elements in $[n]$.*

Proof. Fix a collection $\mathcal{C}$ of k subsets in $\mathcal{S}$ that covers less than $(1-\rho) \cdot n$ elements in $[n]$. The probability that this collection covers U_{smpl} entirely is equal to the probability that none of the $\rho \cdot n$ elements in $[n]$ that are not appearing in $\mathcal{C}$ are sampled in U_{smpl}. Hence,

$$\Pr\left(\mathcal{C} \text{ covers } U_{\mathsf{smpl}}\right) \leq (1-p)^{\rho \cdot n} \leq \exp\left(-\left(16 \cdot k \cdot \log m/(\rho \cdot n)\right) \cdot (\rho \cdot n)\right) \leq 1/m^{8k}$$

Taking a union bound over all $\binom{m}{k} \leq m^k$ possible choices for $\mathcal{C}$ finalizes the result. □

Proof of Lemma 7.6.12. In each of the α iterations, Algorithm 1 implements the sampling in Lemma 7.6.13 with the parameters $k = \widetilde{\mathrm{opt}}$, and $\rho = n^{-1/\alpha}$. Hence, after each iteration, the number of uncovered elements in U reduces to $|U|/n^{1/\alpha}$ w.p. $1 - 1/m^2$. Consequently, by taking a union bound over the $\alpha \leq m$ iterations, after the α iterations, number of uncovered elements reduces to less than 1, hence proving the lemma. □

Proof of Theorem 7.7. We can run Algorithm 1 in parallel for $O(\log n/\varepsilon)$ possible guesses for $\widetilde{\mathrm{opt}}$. By Lemma 7.6.10, the space requirement of this algorithm is $\widetilde{O}(1/\varepsilon) \cdot \widetilde{O}(mn^{1/\alpha}/\varepsilon + n)$ as desired. Moreover, consider the guess: $\mathrm{opt} \leq \widetilde{\mathrm{opt}} \leq (1+\varepsilon) \cdot \widetilde{\mathrm{opt}}$. For this choice, we can apply Lemma 7.6.11 and Lemma 7.6.12 and obtain that the returned solution is an $(\alpha + O(\varepsilon))$-approximation of the optimal set cover. Since the algorithm can make sure that the returned solution is always feasible, returning the smallest set cover among all guesses for $\widetilde{\mathrm{opt}}$ then ensures that the returned answer is an $(\alpha + O(\varepsilon))$ approximation. Re-parameterizing ε by a constant factor, finalizes the proof. □

7.7. Space-Approximation Tradeoff for Multi-pass Coverage

We now prove a space-approximation tradeoff for maximum coverage, formalizing Result 7.3.

Theorem 7.9. *For any $\varepsilon = \omega(1/\sqrt{n})$, $m = \mathrm{poly}(n)$, and $p \geq 1$, any* randomized *algorithm that can make p passes over any collection of m subsets of $[n]$ presented in a random order stream and outputs a $(1+\varepsilon)$-approximation to the optimal value of the maximum coverage problem for $k = 2$ with a sufficiently large constant probability (over the randomness of both the stream order and the algorithm) must use $\widetilde{\Omega}(m/(\varepsilon^2 \cdot p))$ space.*

Similar to previous section, we prove Theorem 7.9 by considering the communication

complexity of the maximum coverage problem: Fix a (sufficiently large) n, $\varepsilon = \omega(1/\sqrt{n})$ and $m = \text{poly}(n)$; MaxCover refers to the communication problem of $(1+\varepsilon)$-approximating the optimal value of the maximum coverage problem with $2m$ sets defined over the universe $[n]$ and parameter $k = 2$, in the two-player communication model.

Our lower bound for MaxCover is obtained by reducing this problem to multiple instances of the *gap-hamming-distance* problem via a similar distribution as $\mathcal{D}_{\mathsf{SC}}$ (using an additional simple gadget).

The Gap-Hamming-Distance Problem. Recall the exact definition of the gap-hamming-distance problem from Section 2.7.2. This problem was originally introduced by [188] and has been studied extensively in the literature (see [88] and references therein). We use the following result on the information complexity of this problem proven in [69].

Lemma 7.7.1 ([69])**.** *Let $\mathcal{U}$ be the uniform distribution on pairs of subsets of $[t]$ (chosen independently); there exists an absolute constant $\delta > 0$ such that for any δ-error protocol π_{GHD} $\mathsf{IC}^{\text{int}}_{\mathcal{U}}(\pi_{\mathsf{GHD}}) = \Omega(t)$.*

For our purpose, we need to consider the following distribution $\mathcal{D}_{\mathsf{GHD}}$ for GHD instead of the uniform distribution. Let $a, b \in [t]$ be two parameters to be determined later[6]. Define:

- $\mathcal{D}^{\mathsf{Y}}_{\mathsf{GHD}}$: distribution of instances $(A, B) \sim \mathcal{U} \mid \Delta(A, B) \geq t/2 + \sqrt{t}, |A| = a, |B| = b$.
- $\mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}$: distribution of instances $(A, B) \sim \mathcal{U} \mid \Delta(A, B) \leq t/2 - \sqrt{t}, |A| = a, |B| = b$.
- $\mathcal{D}_{\mathsf{GHD}} := \frac{1}{2} \cdot \mathcal{D}^{\mathsf{Y}}_{\mathsf{GHD}} + \frac{1}{2} \cdot \mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}$.

We use Lemma 7.7.1 to prove the following result on the information cost of δ-error protocols on the distribution $\mathcal{D}_{\mathsf{GHD}}$, which could be independently useful also. The following lemma can be proven by combining standard ideas to find the values a and b and using the same information odometer argument as Lemma 7.6.7 for disjointness. We omit the proof here and instead refer the interested reader to our paper [27].

Lemma 7.7.2 ([27])**.** *Let $\delta > 0$ be a sufficiently small constant and π_{GHD} be a δ-error protocol for GHD_t on $\mathcal{D}_{\mathsf{GHD}}$ with $\|\pi_{\mathsf{GHD}}\| = 2^{o(t)}$; then, $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}}(\pi_{\mathsf{GHD}}) = \Omega(t)$.*

7.7.1. Communication Complexity of MaxCover

We are now ready to prove a lower bound on the communication complexity of the MaxCover problem. To do so, we propose the following distribution.

[6] Exact values of a and b are not important and are hence only determined in the proof of Lemma 7.7.2.

Distribution $\mathcal{D}_{\mathsf{MC}}$. A hard input distribution for $\mathsf{MaxCover}$.

Notation. Let $t_1 := 1/\varepsilon^2$, $t_2 := 10 \cdot t_1$, $U_1 := [t_1]$ and $U_2 := [t_1 + 1, t_1 + t_2]$.

- For each $i \in [m]$:
  - Let $(A_i, B_i) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}$ for GHD_{t_1} on the universe U_1.
  - Create $C_i, D_i \subseteq U_2$, by assigning each element in U_2 w.p. $1/2$ to C_i and o.w. to D_i.
  - Let $S_i := A_i \cup C_i$ and $T_i := B_i \cup D_i$.
- Pick $\theta \in_R \{0, 1\}$ uniformly at random. If $\theta = 0$, do nothing, otherwise:
  - Sample $i^* \in_R [m]$ uniformly at random.
  - Resample $(A_{i^*}, B_{i^*}) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{GHD}}$ for GHD_{t_1} and redefine S_{i^*} and T_{i^*} as before using the new pair (A_{i^*}, B_{i^*}) (do not change C_i and D_i).
- Let the input to Alice and Bob be $\mathcal{S} := \{S_i\}_{i \in [m]}$ and $\mathcal{T} := \{T_i\}_{i \in [m]}$, respectively.

Define $\mathrm{opt}(\mathcal{S}, \mathcal{T})$ as the value of the optimal solution of the maximum coverage problem (for the parameter $k = 2$) for the instance $(\mathcal{S}, \mathcal{T})$. We argue that $\mathrm{opt}(\mathcal{S}, \mathcal{T})$ differs by a $(1 \pm \varepsilon)$ factor depending on the choice of θ in the distribution.

Lemma 7.7.3. *Assuming $\varepsilon = o(1/\log n)$, there exists a fixed $\tau \in [n]$ such that for any instance $(\mathcal{S}, \mathcal{T}) \sim \mathcal{D}_{\mathsf{MC}}$:*

$$\Pr\left(\mathrm{opt}(\mathcal{S}, \mathcal{T}) \geq (1 + \Theta(\varepsilon)) \cdot \tau \mid \theta = 1\right) = 1 - o(1)$$
$$\Pr\left(\mathrm{opt}(\mathcal{S}, \mathcal{T}) \leq (1 - \Theta(\varepsilon)) \cdot \tau \mid \theta = 0\right) = 1 - o(1)$$

Proof. We first prove that, any $(1+\varepsilon)$-approximate 2-cover in this distribution always has to pick a pair of (S_i, T_i) sets (for some $i \in [m]$). This is achieved by considering the projection of the sets on the universe U_2.

Claim 7.7.4. *W.p. $1 - o(1)$:*

1. *For any $i \in [m]$, $|S_i \cup T_i| \geq t_2$.*
2. *For any $i \neq j \in [m]$, any $Z_i \in \{S_i, T_i\}$, and $Z_j \in \{S_j, T_j\}$, $|Z_i \cup Z_j| \leq (3/4 + 0.2) \cdot t_2$.*

Proof. Part (1) follows immediately from the fact that U_2 is partitioned between S_i and T_i, and that $|U_2| = t_2$. We now prove Part (2). To do so, we prove that $Z_i \cup Z_j$ can only cover (essentially) $3/4$ fraction of U_2 w.h.p and since the rest of $Z_i \cup Z_j$ is a subset of U_1 with $|U_1| \leq 0.1 \cdot t_2$, we get the final result.

For any element $e \in U_2$, define an indicator random variable $X_e \in \{0, 1\}$ where $X_e = 1$ iff $e \in Z_i \cup Z_j$. Since $i \neq j$, the elements in Z_i and Z_j that are in U_2 are chosen independent of

each other, and hence $\Pr X_e = 1 = 1-(1/2)^2 = 3/4$. Define $X := \sum_{e \in U_2} X_e$; we have $\mathbb{E}\, X = 3/4 \cdot t_2$ and since X_e variables are independent, by Chernoff bound, $\Pr X \geq \mathbb{E}\, X + 0.1 \cdot t_2 \leq \exp(-c \cdot t_2) = o(1/m^2)$ (as $t_2 = \omega(\log n)$ and $m = \text{poly}(n)$). The final result now follows from a union bound over all possible ($\leq (2m)^2$) pairs. $\square$ Claim 7.7.4

Now consider a pair (S_i, T_i) for some $i \in [m]$ and note that $|S_i \cup T_i| = |U_2| + |A_i \cup B_i| = t_2 + |A_i \cup B_i|$; hence we can simply focus on $A_i \cup B_i \subseteq U_1$ part of $S_i \cup T_i$. Moreover, we have that, $|A_i \cup B_i| = \frac{1}{2} \cdot (|A_i| + |B_i| + \Delta(A_i, B_i)) = \frac{1}{2} \cdot (a + b + \Delta(A_i, B_i))$ where we used the fact that in the distribution $\mathcal{D}_{\mathsf{GHD}}$, $|A_i| = a$ and $|B_i| = b$ always.

Consequently, whenever $(A_i, B_i) \sim \mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}$, we have,

$$|S_i \cup T_i| = t_2 + |A_i \cup B_i| \leq t_2 + (a+b)/2 + t_1/4 - \sqrt{t_1}/2 = (1 - \Theta(\varepsilon)) \cdot \tau$$

for $\tau := t_2 + (a+b)/2 + t_1/4$. Similarly, whenever $(A_i, B_i) \sim \mathcal{D}^{\mathsf{Y}}_{\mathsf{GHD}}$,

$$|S_i \cup T_i| \geq t_2 + |A_i \cup B_i| \geq t_2 + (a+b)/2 + t_1/4 + \sqrt{t_1}/2 = (1 + \Theta(\varepsilon)) \cdot \tau$$

Combining these bounds with Claim 7.7.4 finalizes the proof. $\square$ Lemma 7.7.3

Having proved Lemma 7.7.3, we can use any $(1+\varepsilon)$-approximation protocol for $\mathsf{MaxCover}$ to determine the parameter θ in the distribution $\mathcal{D}_{\mathsf{MC}}$ (by a simple re-parametrizing of the ε by a constant factor). This allows us to prove the following lemma. The proof is essentially identical to that of Lemma 7.6.6 in Section 7.6.2 and is hence omitted.

Lemma 7.7.5. *Let π_{MC} be a δ-error protocol for $\mathsf{MaxCover}$ on $\mathcal{D}_{\mathsf{MC}}$. There exists a $(\delta + o(1))$-protocol π_{GHD} for GHD_{t_1} on the distribution $\mathcal{D}_{\mathsf{GHD}}$ such that:*

1. $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}}(\pi_{\mathsf{GHD}}) = \frac{O(1)}{m} \cdot \mathsf{IC}^{\text{int}}_{\mathcal{D}_{\mathsf{MC}}}(\pi_{\mathsf{MC}})$.
2. $\|\pi_{\mathsf{GHD}}\| = \|\pi_{\mathsf{MC}}\|$.

Theorem 7.10. *For sufficiently small constant $\delta > 0$, and $\omega(1/\sqrt{n}) \leq \varepsilon \leq o(1/\log n)$, and $m = \text{poly}(n)$, communication complexity of $\mathsf{MaxCover}$ is $\mathsf{R}_\delta(\mathsf{MaxCover}) = \Omega(m/\varepsilon^2)$.*

Proof. Suppose there exists a δ-error protocol π_{MC} for $\mathsf{MaxCover}$ on $\mathcal{D}_{\mathsf{MC}}$ for a sufficiently small constant δ with $\|\pi_{\mathsf{GHD}}\| = o(m/\varepsilon^2)$; by Proposition 2.8.3, $\mathsf{IC}^{\text{int}}_{\mathcal{D}_{\mathsf{MC}}}(\pi_{\mathsf{MC}}) = o(m/\varepsilon^2)$ as well. Hence, by Lemma 7.7.5, we obtain a $(\delta + o(1))$-error protocol π_{GHD} for GHD_{t_1} on $\mathcal{D}_{\mathsf{GHD}}$ with $\|\pi_{\mathsf{GHD}}\| = o(m/\varepsilon^2)$ and $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}}(\pi_{\mathsf{GHD}}) = o(1/\varepsilon^2) = o(t_1)$. However, since $\|\pi_{\mathsf{GHD}}\| = o(m/\varepsilon^2) = 2^{o(t_1)}$ as $m = \text{poly}(n)$ and $t_1 = \omega(\log n)$, we can now apply Lemma 7.7.2 and argue that $\mathsf{IC}^{\text{int}}_{\mathcal{D}^{\mathsf{N}}_{\mathsf{GHD}}}(\pi_{\mathsf{GHD}})$ is $\Omega(t_1)$ (by taking δ smaller than the bounds in the Lemma 7.7.2); a contradiction with the information cost of π_{GHD} obtained by Lemma 7.7.5. $\square$

Chapter 8
Submodular Maximization in the Distributed Communication Model

In this chapter, we study the general problem of submodular maximization subject to cardinality constraint and its canonical example, the maximum coverage problem. Throughout this section, we mostly focus on the maximum coverage problem in the distributed communication model but our results have several interesting ramifications for general submodular maximization problems as well as streaming and MPC models studied in this thesis. Recall the definition of maximum coverage and the distributed communication model from Sections 2.4 and 1.1.2, respectively. In order to avoid confusion, throughout this chapter, we use k to denote the target number of sets in the maximum coverage problem and use p (instead of the usual k) to denote the number of machines in the distributed model. The materials in this chapter are based on [31].

Previous results for maximum coverage in the distributed model can be divided into two main categories: one on hand, we have *communication efficient* protocols that only need $\widetilde{O}(n)$ communication (n denotes the size of the universe) and achieve a constant factor approximation, but require a *large* number of rounds of $\Omega(p)$ [40, 247]. On the other hand, we have *round efficient* protocols that achieve a constant factor approximation in $O(1)$ rounds of communication, but incur a *large* communication cost $k \cdot m^{\Omega(1)}$ [223] (m denotes the number of the sets). This state-of-the-affairs, namely, communication efficient protocols that require a large number of rounds, or round efficient protocols that require a large communication cost, raises the following natural question:

> *Does there exist a truly efficient distributed protocol for maximum coverage, that is, a protocol that simultaneously achieves $\widetilde{O}(n)$ communication cost, $O(1)$ round complexity, and gives a constant factor approximation?*

We refute the possibility of this optimistic scenario by presenting a *tight tradeoff* between the three main measures of efficiency for the distributed coverage problem: the approximation ratio, the communication cost, and the number of rounds. As a corollary of our results, we also obtain the first multi-pass dynamic streaming lower bounds for any optimization problem, as well as a simple MPC algorithm for submodular maximization subject to cardinality constraint that matches the best known bounds in the literature.

HighLights of Our Contributions

In this chapter, we will establish:

- A new framework for establishing communication complexity lower bounds for bounded-round protocols in the multiparty communication model (Section 8.4).
- Tight upper and lower bounds on the round-complexity of the distributed maximum coverage and submodular maximization subject to cardinality constraint (Sections 8.5 and 8.6).
- Dynamic streaming and MPC algorithms and lower bounds for maximum coverage and submodular maximization subject to cardinality constraint (Section 8.7).

8.1. Background

A common paradigm for designing scalable algorithms for problems on massive data sets is to distribute the computation by partitioning the data across multiple machines interconnected via a communication network. The machines can then jointly compute a function on the union of their inputs by exchanging messages. A well-studied and important case of this paradigm is the distributed communication model we introduced in Section 1.1.2. In this model, the computation proceeds in rounds, and in each round, all machines simultaneously send a message to a central coordinator who then communicates back to all machines a summary to guide the computation for the next round. At the end of the last round, the coordinator outputs the answer. Main measures of efficiency in this model are the communication cost and the round complexity.

The distributed coordinator model (and the closely related message-passing model[1]) has been studied extensively in recent years (see, e.g., [268, 67, 302, 303, 304], and references therein). Traditionally, the focus in this model has been on optimizing the communication cost and round complexity issues have been ignored. However, in recent years, motivated by application to big data analysis such as MapReduce computation, there have been a growing interest in obtaining round efficient protocols for various problems in this model (see, e.g., [10, 209, 186, 165, 249, 112, 35, 166, 30]).

Submodular maximization subject to cardinality constraint and its illustrative example, maximum coverage, have been studied extensively in models of computation for massive datasets including in distributed communication model [186, 250, 249, 112], MPC model [99, 63, 223, 113, 51], and the streaming model [281, 87, 40, 50, 247, 129, 261, 38, 50, 94, 247, 27] (this is by no means a comprehensive list of previous results).

Prior to our work, the only known lower bound for distributed maximum coverage in particular and submodular maximization in general was due to McGregor and Vu [247] who showed an $\Omega(m)$ communication lower bound for any protocol that achieves a better than $\left(\frac{e}{e-1}\right)$-approximation (regardless of number of rounds and even if the input is *randomly*

[1] In absence of any restriction on round complexity, these two models are equivalent; see Section 1.1.2.

distributed) (see also our own results in Chapter 7). Indyk *et al.* [186] also showed that no composable coreset (a restricted family of single round protocols) can achieve a better than $\widetilde{\Omega}(\sqrt{k})$ approximation without communicating essentially the whole input (which is known to be tight [112]). However, no super constant lower bounds on approximation ratio were known for this problem for arbitrary protocols even for one round of communication.

In the *dynamic (set) streaming* model, at each step, either a new set is inserted or a previously inserted set is deleted from the stream. The goal is to solve the maximum coverage problem on the sets that are present at the end of the stream. A *semi-streaming* algorithm is allowed to make one or a small number of passes over the stream and use only $O(n \cdot \text{poly}\{\log m, \log n\})$ space to process the stream and compute the answer. The streaming setting for the maximum coverage problem and the closely related set cover problem has been studied extensively in recent years [281, 108, 38, 87, 128, 116, 40, 174, 90, 32, 50, 94, 247, 27, 129] (see Chapter 7 for a summary of these works). Previous work considered this problem in *insertion-only* streams and more recently in the *sliding window* model, however, no non-trivial results were known for this problem in dynamic streams.

8.2. Our Results and Techniques

Our first result is a *negative* resolution of our motivating question on existence of communication, round, and approximation efficient distributed algorithms for maximum coverage. In particular, we show that,

Result 8.1. *For any integer $r \geq 1$, any r-round protocol for distributed maximum coverage either incurs $k \cdot m^{\Omega(1/r)}$ communication per machine or has an approximation factor of $k^{\Omega(1/r)}$.*

Prior to our work, no super constant lower bounds on approximation ratio were known for this problem for arbitrary protocols even for one round of communication. Our result on the other hand implies that to achieve any constant factor approximation with any $O(n^c)$ communication protocol (for a fixed constant $c > 0$), $\Omega\left(\frac{\log k}{\log \log k}\right)$ rounds of communication are required.

In establishing Result 8.1, we introduce a general framework for proving communication complexity lower bounds for *bounded round* protocols in the distributed coordinator model. This framework, formally introduced in Section 8.4, captures many of the existing multi-party communication complexity lower bounds in the literature for bounded-round protocols including [119, 220, 35, 33] (for one round a.k.a simultaneous protocols), and [19, 25] (for multi-round protocols), including our own lower bounds in Chapters 3. We believe our framework will prove useful for establishing distributed lower bound results for other problems, and is thus interesting in its own right.

We complement Result 8.1 by giving protocols that show that its bounds are *tight*.

Result 8.2. *For any integer $r \geq 1$, there exist r-round protocols that achieve:*

1. *an (almost) $\left(\frac{e}{e-1}\right)$-approximation with $k \cdot m^{O(1/r)}$ communication per machine,*
2. *an $O(r \cdot k^{1/r+1})$-approximation with $\widetilde{O}(n)$ communication per machine.*

Results 8.1 and 8.2 together provide a near complete understanding of the tradeoff between the approximation ratio, the communication cost, and the round complexity of protocols for the distributed maximum coverage problem for any fixed number of rounds.

The first protocol in Result 8.2 is quite general in that it works for maximizing any monotone submodular function subject to a cardinality constraint. Previously, it was known how to achieve a 2-approximation distributed algorithm for this problem with $m^{O(1/r)}$ communication and r rounds of communication [223]. However, the previous best $\left(\frac{e}{e-1}\right)$-approximation distributed algorithm for this problem with *sublinear* in m communication due to Kumar *et al.* [223] requires at least $\Omega(\log n)$ rounds of communication. As noted above, the $\left(\frac{e}{e-1}\right)$ is information theoretically the best approximation ratio possible for any protocol that uses sublinear in m communication [247].

The second protocol in Result 8.2 is however tailored heavily to the maximum coverage problem. Previously, it was known that an $O(\sqrt{k})$ approximation can be achieved via $\widetilde{O}(n)$ communication [112] per machine, but no better bounds were known for this problem in multiple rounds under poly(n) communication cost. It is worth noting that since an adversary may assign all sets to a single machine, a communication cost of $\widetilde{O}(n)$ is essentially best possible bound. We now elaborate on some applications of our results.

Dynamic streams. Our Results 8.1 and 8.2 imply the first upper and lower bounds for maximum coverage in dynamic streams. Result 8.1 together with the connection between distributed lower bounds and dynamic streams [14] (Proposition 2.7.11) proves that any semi-streaming algorithm for maximum coverage in dynamic streams that achieves any *constant approximation* requires $\Omega\left(\frac{\log n}{\log \log n}\right)$ passes over the stream. This is in sharp contrast with insertion-only streams in which semi-streaming algorithms can achieve (almost) 2-approximation in only a single pass [40] or (almost) $\left(\frac{e}{e-1}\right)$-approximation in a constant number of passes [247] (constant factor approximations are also known in the sliding window model [94, 129]). To our knowledge, this is the first *multi-pass* dynamic streaming lower bound that is based on the characterization of [14]. Moreover, as maximum coverage is a special case of submodular maximization (subject to cardinality constraint), our lower bound immediately extends to this problem and settles an open question of [129].

We complement this result by showing that one can implement the first algorithm in Result 8.2 using proper *linear sketches* in dynamic streams, which imply an (almost) $\left(\frac{e}{e-1}\right)$-

approximation semi-streaming algorithm for maximum coverage (and monotone submodular maximization) in $O(\log m)$ passes. As a simple application of this result, we can also obtain an $O(\log n)$-approximation semi-streaming algorithm for the set cover problem in dynamic stream that requires $O(\log m \cdot \log n)$ passes over the stream.

MPC model. Proving round complexity lower bounds in the MapReduce framework turns out to be a challenging task (see Section 1.1.3) As a result, most previous work on lower bounds concerns either communication cost (in a fixed number of rounds) or specific classes of algorithms (for round lower bounds); see, e.g., [3, 53, 269, 189] (see [277] for more details). Our results contribute to the latter line of work by characterizing the power of a large family of MapReduce algorithms for maximum coverage.

Many existing techniques for MapReduce algorithms utilize the following paradigm which we call the *sketch-and-update* approach: each machine sends a summary of its input, i.e., a sketch, to a *single* designated machine which processes these sketches and computes a single combined sketch; the original machines then receive this combined sketch and update their sketch computation accordingly; this process is then continued on the updated sketches. Popular algorithmic techniques belonging to this framework include composable coresets (e.g., [40, 43, 49, 186]), the filtering method (e.g., [225]), linear-sketching algorithms (e.g., [10, 209, 9]), and the sample-and-prune technique (e.g., [223, 184]), among many others.

We use Result 8.1 to prove a lower bound on the power of this approach for solving maximum coverage in the MapReduce model. We show that any MapReduce algorithm for maximum coverage in the sketch-and-update framework that uses $s = m^{\delta}$ memory per machine requires $\Omega(\frac{1}{\delta})$ rounds of computation. Moreover, both our algorithms in Result 8.2 belong to the sketch-and-update framework and can be implemented in the MapReduce model. In particular, the round complexity of our first algorithm for monotone submodular maximization (subject to cardinality constraint) in Result 8.2 matches the best known algorithm of [113] with the benefit of using sublinear communication (the algorithm of [113], in each round, incurs a linear (in input size) communication cost). We remark that the algorithm in [113] is however more general in that it supports a larger family of constraints beside the cardinality constraint we study in this paper.

8.3. Technical Overview

Lower bounds (Result 8.1). Let us start by sketching our proof for simultaneous protocols. We provide each machine with a collection of sets from a family of sets with small pairwise intersection such that *locally*, i.e., from the perspective of each machine, all these sets look alike. At the same time, we ensure that *globally*, one set in each machine is *special*; think of a special set as covering a *unique* set of elements across the machines while all other

sets are mostly covering a set of *shared* elements. The proof now consists of two parts: (*i*) use the simultaneity of the communication to argue that as each machine is oblivious to identity of its special set, it cannot convey enough information about this set using limited communication, and (*ii*) use the bound on the size of the intersection between the sets to show that this prevents the coordinator to find a good solution.

The strategy outlined above is in fact at the core of many existing lower bounds for simultaneous protocols in the coordinator model including [119, 220, 35, 33] (a notable exception is the lower bound of [33] on estimating matching size in *sparse* graphs). For example, to obtain the hard input distributions in [220, 35] for the maximum matching problem, we just need to switch the sets in the small intersecting family above with induced matchings in a Ruzsa-Szemerédi graph [279] (see also [18] for more details on these graphs). The first part of the proof that lower bounds the communication cost required for finding the special induced matchings (corresponding to special sets above), remains quite similar; however, we now need an entirely different argument for proving the second part, i.e., the bound obtained on the approximation ratio. This observation raises the following question: can we somehow "automate" the task of proving a communication lower bound in the arguments above so that one can focus solely on the second part of the argument, i.e., proving the approximation lower bound subject to each machine not being able to find its special entity, e.g., sets in the coverage problem and induced matchings in the maximum matching problem?

We answer this question in the affirmative by designing a framework for proving communication lower bounds of the aforementioned type. We design an abstract hard input distribution using the ideas above and prove a general communication lower bound in this abstraction. This reduces the task of proving a communication lower bound for any specific problem to designing suitable combinatorial objects that roughly speaking enforce the importance of "special entities" discussed above. We emphasize that this second part may still be a non-trivial challenge; for instance, lower bounds for matchings in [220, 35] rely on Ruzsa-Szemerédi graphs to prove this part. Nevertheless, automating the task of proving a communication lower bound in our framework allows one to focus solely on a combinatorial problem and entirely bypass the communication lower bounds argument.

We further extend our framework to multi round protocols by building on the recent multi-party round elimination technique of [19] and its extension in [25]. At a high level, in the hard instances of r-round protocols, each machine is provided with a collection of instances of the same problem but on a "lower dimension", i.e., defined on a smaller number of machines and input size. One of these instances is a special one in that it needs to be solved by the machines in order to solve the original instance. Again, using the simultaneity of the communication in one round, we show that the first round of communication cannot

reveal enough information about this special instance and hence the machines need to solve the special instance in only $r-1$ rounds of communication, which is proven to be hard inductively. Using the abstraction in our framework allows us to solely focus on the communication aspects of this argument, independent of the specifics of the problem at hand. This allows us to provide a more direct and simpler proof than [19, 25], which is also applicable to a wider range of problems (the results in [19, 25] are for the setting of combinatorial auctions). However, although simpler than [19, 25], this proof is still far from being simple - indeed, it requires a delicate information-theoretic argument (see Section 8.4 for further details). This complexity of proving a multi-round lower bound in this model is in fact another motivation for our framework. To our knowledge, the only previous lower bounds specific to bounded round protocols in the coordinator model are those of [19, 25]; we hope that our framework facilitates proving such lower bounds in this model (understanding the power of bounded round protocols in this model is regarded as an interesting open question in the literature; see, e.g., [303]).

Finally, we prove the lower bound for maximum coverage using this framework by designing a family of sets which we call *randomly nearly disjoint*; roughly speaking the sets in this family have the property that any suitably small *random* subset of one set is essentially disjoint from any other set in the family. A reader familiar with [90] may realize that this definition is similar to the *edifice* set-system introduced in [90]; the main difference here is that we need every *random* subsets of each set in the family to be disjoint from other sets, as opposed to a *pre-specified* collection of sets as in edifices [90]. As a result, the algebraic techniques of [90] do not seem suitable for our purpose and we prove our results using different techniques. The lower bound then follows by instantiating the hard distribution in our framework with this family for maximum coverage and proving the approximation lower bound.

Upper bounds (Result 8.2). We achieve the first algorithm in Result 8.2, namely an $\left(\frac{e}{e-1}\right)$-approximation algorithm for maximum coverage (and submodular maximization), via an implementation of a thresholding greedy algorithm (see, e.g., [41, 90]) in the distributed setting using the sample-and-prune technique of [223] (a similar thresholding greedy algorithm was used recently in [247] for streaming maximum coverage). The main idea in the sample-and-prune technique is to sample a collection of sets from the machines in each round and send them to the coordinator who can build a partial greedy solution on those sets; the coordinator then communicates this partial solution to each machine and in the next round the machines only sample from the sets that can have a substantial marginal contribution to the partial greedy solution maintained by the coordinator. Using a different greedy algorithm and a more careful choice of the threshold on the necessary marginal contribution from each set, we show that an $\left(\frac{e}{e-1}\right)$-approximation can be obtained in con-

stant number of rounds and sublinear communication (as opposed to the original approach of [223] which requires $\Omega(\log n)$ rounds).

The second algorithm in Result 8.2, namely a $k^{O(1/r)}$-approximation algorithm for any number of rounds r, however is more involved and is based on a new iterative sketching method specific to the maximum coverage problem. Recall that in our previous algorithm the machines are mainly "observers" and simply provide the coordinator with a sample of their input; our second algorithm is in some sense on the other extreme. In this algorithm, each machine is responsible for computing a suitable sketch of its input, which roughly speaking, is a collection of sets that tries to "represent" each optimal set in the input of this machine. The coordinator is also maintaining a greedy solution that is updated based on the sketches received from each machine. The elements covered by this collection are shared by the machines to guide them towards the sets that are "misrepresented" by the sketches computed so far, and the machines update their sketches for the next round accordingly. We show that either the greedy solution maintained by the coordinator is already a good approximation or the final sketches computed by the machines are now a good representative of the optimal sets and hence contain a good solution.

8.4. A Framework for Proving Distributed Lower Bounds

We introduce a general framework for proving communication complexity lower bounds for *bounded round* protocols in the distributed coordinator model. Consider a *decision* problem[2] $\mathcal{P}$ defined by the family of functions $\mathcal{P}_s : \{0,1\}^s \to \{0,1\}$ for any integer $s \geq 1$; we refer to s as *size* of the problem and to $\{0,1\}^s$ as its *domain*. Note that $\mathcal{P}_s$ can be a partial function, i.e., not necessarily defined on its whole domain. An instance I of problem $\mathcal{P}_s$ is simply a binary string of length s. We say that I is a Yes instance if $\mathcal{P}_s(I) = 1$ and is a No instance if $\mathcal{P}_s(I) = 0$. For example, $\mathcal{P}_s$ can denote the decision version of the maximum coverage problem over m sets and n elements with parameter k (in which case s would be a fixed function of m, n, and k depending on the representation of the input) such that there is a relatively large gap (as a function of, say, k) between the value of optimal solution in Yes and No instances. We can consider the problem $\mathcal{P}_s$ in the distributed model, where we distribute each instance between the players. The distributed coverage problem for instance, can be modeled by partitioning the sets in the instances of $\mathcal{P}_s$ across players.

To prove a communication lower bound for some problem $\mathcal{P}$, one typically needs to design a hard input distribution $\mathcal{D}$ on instances of the problem $\mathcal{P}$, and then show that distinguishing between the Yes and No cases in instances sampled from $\mathcal{D}$, with some sufficiently large probability, requires large communication. Such a distribution inevitably depends on the specific problem $\mathcal{P}$ at hand. We would like to *abstract out* this dependence

[2]While we present our framework for decision problems, it also extends to *search* problems; see [31].

to the underlying problem and design a *template* hard distribution for any problem $\mathcal{P}$ using this abstraction. Then, to achieve a lower bound for a particular problem $\mathcal{P}$, one only needs to focus on the problem specific parts of this template and *design* them according to the problem $\mathcal{P}$ at hand. We emphasize that obviously we are not going to prove a communication lower bound for every possible distributed problem; rather, our framework reduces the problem of proving a communication lower bound for a problem $\mathcal{P}$ to designing appropriate problem-specific *gadgets* for $\mathcal{P}$, which determine the strength of the lower bound one can ultimately prove using this framework. With this plan in mind, we now describe a high level overview of our framework.

8.4.1. A High Level Overview of the Framework

Consider any decision problem $\mathcal{P}$; we construct a recursive family of distributions $\mathcal{D}_0, \mathcal{D}_1, \ldots$ where $\mathcal{D}_r$ is a hard input distribution for r-round protocols of $\mathcal{P}_{s_r}$, i.e., for instances of size s_r of the problem $\mathcal{P}$, when the input is partitioned between p_r players. Each instance in $\mathcal{D}_r$ is a careful "combination" of many sub-instances of problem $\mathcal{P}_{s_{r-1}}$ over different subsets of p_{r-1} players, which are sampled (essentially) from $\mathcal{D}_{r-1}$. We ensure that a small number of these sub-instances are "special" in that to solve the original instance of $\mathcal{P}_{s_r}$, at least one of these instances of $\mathcal{P}_{s_{r-1}}$ (over p_{r-1} players) needs to be solved necessarily. We "hide" the special sub-instances in the input of players in a way that locally, no player is able to identify them and show that the first round of communication in any protocol with a small communication is spent only in identifying these special sub-instances. We then inductively show that as solving the special instance is hard for $(r-1)$-round protocols, the original instance must be hard for r-round protocols as well.

We now describe this distribution in more detail. The p_r players in the instances of distribution $\mathcal{D}_r$ are partitioned into g_r groups $P_1, \ldots, P_{g_r}$, each of size p_{r-1} (hence $g_r = p_r/p_{r-1}$). For every group $i \in [g_r]$ and every player $q \in P_i$, we create w_r instances $I_1^i, \ldots, I_{w_r}^i$ of the problem $\mathcal{P}_{s_{r-1}}$ sampled from the distribution $\mathcal{D}_{r-1}$. The domain of each instance I_j^i is the same across all players in P_i and is different (i.e., disjoint) between any two $j \neq j' \in [w_r]$; we refer to w_r as the *width parameter*. The next step is to *pack* all these instances into a single instance $I^i(q)$ for the player q; this is one of the places that we need a problem specific gadget, namely a *packing function*[3] that can pack w_r instances of problem $\mathcal{P}_{s_{r-1}}$ into a single instance of problem $\mathcal{P}_{s'_r}$ for some $s'_r \geq s_r$. We postpone the formal description of the packing functions to the next section, but roughly speaking, we require each player to be able to construct the instance $I^i(q)$ from the instances $I_1^i, \ldots, I_{w_r}^i$ and vice versa. As

[3]For a reader familiar with previous work in [33, 19, 25], we note that a similar notion to a packing function is captured via a collection of disjoint blocks of vertices in [19] (for finding large matchings), Ruzsa-Szemerédi graphs in [33] (for estimating maximum matching size), and a family of small-intersecting sets in [25] (for finding good allocations in combinatorial auctions). In this work, we use the notion of randomly nearly disjoint set-systems defined in Section 8.5.1.

such, even though each player is given as input a single instance I^i, we can think of each player as conceptually "playing" in w_r different instances $I^i_1, \ldots, I^i_{w_r}$ of $\mathcal{P}_{s_{r-1}}$ instead.

In each group $i \in [g_r]$, one of the instances, namely $I^i_{j^*}$ for $j^* \in [w_r]$, is the *special instance* of the group: if we combine the inputs of players in P_i on their special instance $I^i_{j^*}$, we obtain an instance which is sampled from the distribution $\mathcal{D}_{r-1}$. On the other hand, all other instances are *fooling instances*: if we combine the inputs of players in P_i on their instance I^i_j for $j \neq j^*$, the resulting instance is *not* sampled from $\mathcal{D}_{r-1}$; rather, it is an instance created by picking the input of each player *independently* from the corresponding marginal of $\mathcal{D}_{r-1}$ ($\mathcal{D}_{r-1}$ is *not* a product distribution, thus these two distributions are not identical). Nevertheless, by construction, each player is oblivious to this difference and hence is unaware of which instance in the input is the special instance (since the marginal distribution of a player's input is identical under the two distributions above).

Finally, we need to combine the instances $I^1, \ldots, I^{g_r}$ to create the final instance I. To do this, we need another problem specific gadget, namely a *relabeling function*. Roughly speaking, this function takes as input the index j^*, i.e., the index of the special instances, and instances $I^1, \ldots, I^{g_r}$ and create the final instance I, while "prioritizing" the role of special instances in I. By prioritizing we mean that in this step, we need to ensure that the value of $\mathcal{P}_{s_r}$ on I is the same as the value of $\mathcal{P}_{s_{r-1}}$ on the special instances. At the same time, we also need to ensure that this additional relabeling does not reveal the index of the special instance to each individual player, which requires a careful design depending on the problem at hand.

The above family of distributions is parameterized by the sequences $\{s_r\}$ (size of instances), $\{p_r\}$ (number of players), and $\{w_r\}$ (the width parameters), plus the packing and relabeling functions. Our main result in this section is that if these sequences and functions satisfy some natural conditions (similar to what discussed above), then any r-round protocol for the problem $\mathcal{P}_{s_r}$ on the distribution $\mathcal{D}_r$ requires $\Omega_r(w_r)$ communication.

We remark that while we state our communication lower bound only in terms of w_r, to obtain any interesting lower bound using this technique, one needs to ensure that the width parameter w_r is relatively large in the size of the instance s_r; this is also achieved by designing suitable packing and labeling functions (as well as a suitable representation of the problem). However, as "relatively large" depends heavily on the problem at hand, we do not add this requirement to the framework explicitly.

Further discussions on extensions of this framework and its connection to previous work is also presented in our paper [31].

8.4.2. The Formal Description of the Framework

We now describe our framework formally. As stated earlier, to use this framework for proving a lower bound for any specific problem $\mathcal{P}$, one needs to define appropriate problem-specific gadgets. These gadgets are functions that map multiple instances of $\mathcal{P}_s$ to a single instance $\mathcal{P}_{s'}$ for some $s' \geq s$. The exact application of these gadgets would become clear shortly in the description of our hard distribution for the problem $\mathcal{P}$.

Definition 8.1 (Packing Function)**.** *For integers $s' \geq s \geq 1$ and $w \geq 1$, we refer to a function σ which maps any tuple of instances $(I_1, \ldots, I_w)$ of $\mathcal{P}_s$ to a single instance I of $\mathcal{P}_{s'}$ as a* packing function *of* width w*.*

Definition 8.2 (Labeling Family)**.** *For integers $s'' \geq s' \geq 1$ and $g \geq 1$, we refer to a family of functions $\Phi = \{\phi_i\}$, where each ϕ_i is a function that maps any tuple of instances $(I^1, \ldots, I^g)$ of $\mathcal{P}_{s'}$ to a single instance I of $\mathcal{P}_{s''}$ as a* g-labeling family*, and to each function in this family, as a* labeling function*.*

We start by designing the following recursive family of hard distributions $\{\mathcal{D}_r\}_{r\geq 0}$, parametrized by sequences $\{p_r\}_{r\geq 0}$, $\{s_r\}_{r\geq 0}$, and $\{w_r\}_{r\geq 0}$. We require $\{p_r\}_{r\geq 0}$ and $\{s_r\}_{r\geq 0}$ to be *increasing* sequences and $\{w_r\}_{r\geq 0}$ to be *non-increasing*. In two places marked in the distribution, we require one to design the aforementioned problem-specific gadgets for the distribution.

Distribution $\mathcal{D}_r$: A template hard distribution for r-round protocols of $\mathcal{P}$ for any $r \geq 1$.

Parameters: p_r: number of players, s_r: size of the instance, w_r: width parameter, σ_r: packing function, and Φ_r: labeling family.

1. Let P be the set of p_r players and define $g_r := \frac{p_r}{p_{r-1}}$; partition the players in P into g_r *groups* $P_1, \ldots, P_{g_r}$ each containing p_{r-1} players.
2. ***Design*** a packing function σ_r of width w_r which maps w_r instances of $\mathcal{P}_{s_{r-1}}$ to an instance of $\mathcal{P}_{s'_r}$ for some $s_{r-1} \leq s'_r \leq s_r$.
3. Pick an instance $I^\star_r \sim \mathcal{D}_{r-1}$ over the set of players $[p_{r-1}]$ and domain of size s_{r-1}.
4. For each group P_i for $i \in [g_r]$:
   (a) Pick an index $j^* \in [w_r]$ *uniformly at random* and create w_r instances $I^i_1, \ldots, I^i_{w_r}$ of problem $\mathcal{P}_{s_{r-1}}$ as follows:
      i. Each instance I^i_j for $j \in [w_r]$ is over the players P_i and domain $D^i_j = \{0,1\}^{s_{r-1}}$.
      ii. For index $j^* \in [w_r]$, $I^i_{j^*} = I^\star_r$ by mapping (arbitrarily) $[p_{r-1}]$ to P_i and domain of $I^\star_r$ to $D^i_{j^*}$.

iii. For any other index $j \neq j^*$, $I^i_j \sim \mathcal{D}'_{r-1} := \otimes_{q \in P_i} \mathcal{D}_{r-1}(q)$, i.e., the *product of marginal distribution* of the input to each player $q \in P_i$ in $\mathcal{D}_{r-1}$.

(b) Map all the instances $I^i_1, \ldots, I^i_{w_r}$ to a single instance I^i using the function σ_r.

5. ***Design*** a g_r*-labeling family* Φ_r which maps g_r instances of $\mathcal{P}_{s'_r}$ to a *single* instance $\mathcal{P}_{s_r}$.
6. Pick a labeling function ϕ from Φ *uniformly at random* and map the g_r instances $I^1, \ldots, I^{g_r}$ of $\mathcal{P}_{s'_r}$ to the output instance I of $\mathcal{P}_{s_r}$ using ϕ.
7. The input to each player $q \in P_i$ in the instance I, for any $i \in [g_r]$, is the input of player q in the instance I^i, *after* applying the mapping ϕ to map I^i to I.

We remark that in the above distribution, the "variables" in each instance sampled from $\mathcal{D}_r$ are the instances $I^i_1, \ldots, I^i_{w_r}$ for all groups $i \in [g_r]$, the index $j^* \in [w]$, and both the choice of labeling family Φ_r and the labeling function ϕ. On the other hand, the "constants" across all instances of $\mathcal{D}_r$ are parameters p_r, s_r, and w_r, the choice of grouping $P_1, \ldots, P_{g_r}$, and the packing function σ_r.

To complete the description of this recursive family of distributions, we need to explicitly define the distribution $\mathcal{D}_0$ between p_0 players over $\{0,1\}^{s_0}$. We let $\mathcal{D}_0 := \frac{1}{2} \cdot \mathcal{D}_0^{\mathsf{Yes}} + \frac{1}{2} \cdot \mathcal{D}_0^{\mathsf{No}}$, where $\mathcal{D}_0^{\mathsf{Yes}}$ is a distribution over Yes instances of $\mathcal{P}_{s_0}$ and $\mathcal{D}_0^{\mathsf{No}}$ is a distribution over No instances. The choice of distributions $\mathcal{D}_0^{\mathsf{Yes}}$ and $\mathcal{D}_0^{\mathsf{No}}$ are again *problem-specific*.

We start by describing the main properties of the packing and labeling functions that are required for our lower bound. For any player $q \in P_i$, define $I^i(q) := (I^i_1(q), \ldots, I^i_{w_r}(q))$, where for any $j \in [w_r]$, $I^i_j(q)$ denotes the input of player q in the instance I^i_j. We require the packing and labeling functions to be *locally computable* defined as follows.

Definition 8.3 (Locally computable)**.** *We say that the packing function σ_r and the labeling family Φ_r are* locally computable *iff any player $q \in P_i$ for $i \in [g_r]$, can compute the mapping of $I^i(q)$ to the final instance I, locally, i.e., only using σ_r, the sampled labeling function $\phi \in \Phi_r$, and input $I^i(q)$.*

We use ϕ_q to denote the *local mapping* of player $q \in P_i$ for mapping $I^i(q)$ to I; since σ_r is fixed in the distribution $\mathcal{D}_r$, across different instances sampled from $\mathcal{D}_r$, ϕ_q is only a function of ϕ. Notice that the input to each player $q \in P_i$ is *uniquely* determined by $I^i(q)$ and ϕ_q.

Inside each instance I sampled from $\mathcal{D}_r$, there exists a unique *embedded* instance $I^\star_r$ which is sampled from $\mathcal{D}_{r-1}$. Moreover, this instance is essentially "copied" g_r times, once in each instance $I^i_{j^*}$ for each group P_i. We refer to the instance $I^\star_r$ as well as its copies

$I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ as *special instances* and to all other instances as *fooling instances*. We require the packing and labeling functions to be *preserving*, defined as,

Definition 8.4 (γ-Preserving)**.** *We say that the packing function and the labeling family are* γ-preserving *for a parameter* $\gamma \in (0,1)$*, iff*

$$\Pr_{I \sim \mathcal{D}_r} \left(\mathcal{P}_{s_r}(I) = \mathcal{P}_{s_{r-1}}(I^\star_r) \right) \geq 1 - \gamma.$$

In other words, the value of $\mathcal{P}_{s_r}$ *on an instance* I *should be equal to the value of* $\mathcal{P}_{s_{r-1}}$ *on the embedded special instance* $I^\star_r$ *of* I *w.p.* $1 - \gamma$*.*

Recall that the packing function σ_r is a deterministic function that depends only on the distribution $\mathcal{D}_r$ itself and not any specific instance (and hence the underlying special instances); on the other hand, the preserving property requires the packing and labeling functions to somehow "prioritize" the special instances over the fooling instances (in determining the value of the original instance). To achieve this property, the labeling family is allowed to vary based on the specific instance sampled from the distribution $\mathcal{D}_r$. However, we need to limit the dependence of the labeling family to the underlying instance, which is captured through the definition of *obliviousness* below.

Definition 8.5. *We say that the labeling family* Φ_r *is* oblivious *iff it satisfies the following properties:*

1. *The only variable in* $\mathcal{D}_r$ *which* Φ_r *can depend on is* $j^* \in [w_r]$ *(it can depend arbitrarily on the constants in* $\mathcal{D}_r$*).*
2. *For any* $q \in P$*, the local mapping* ϕ_q *and* j^* *are* independent *of each other in* $\mathcal{D}_r$*.*

Intuitively speaking, Condition (1) above implies that a function $\phi \in \Phi_r$ can "prioritize" the special instances based on the index j^*, but it cannot use any further knowledge about the special or fooling instances. For example, one may be able to use ϕ to distinguish special instances from other instances, i.e., determine j^*, but would not be able to infer whether the special instance is a Yes instance or a No one only based on ϕ. Condition (2) on the other hand implies that for each player q, no information about the special instance is revealed by the local mapping ϕ_q. This means that given the function ϕ_q (and not ϕ as a whole), one is not able to determine j^*.

Finally, we say that the family of distributions $\{\mathcal{D}_r\}$ is a *γ-hard recursive family*, iff (i) it is parameterized by increasing sequences $\{p_r\}$ and $\{s_r\}$, and non-increasing sequence $\{w_r\}$, and (ii), the packing and labeling functions in the family are locally computable, γ-preserving, and oblivious. We are now ready to present our main theorem of this section.

Theorem 8.3. *Let $R \geq 1$ be an integer and suppose $\{\mathcal{D}_r\}_{r=0}^{R}$ is a γ-hard recursive family for some $\gamma \in (0,1)$; for any $r \leq R$, any r-round protocol for $\mathcal{P}_{s_r}$ on $\mathcal{D}_r$ which errs w.p. at most $1/3 - r \cdot \gamma$ requires $\Omega(w_r/r^4)$ total communication.*

8.4.3. Correctness of the Framework: Proof of Theorem 8.3

We first set up some notation. For any r-round protocol π and any $\ell \in [r]$, we use $\Pi_\ell := (\Pi_{\ell,1}, \ldots, \Pi_{\ell,p_r})$ to denote the random variable for the transcript of the message communicated by each player in round ℓ of π. We further use Φ (resp. Φ_q) to denote the random variable for ϕ (resp. local mapping ϕ_q) and J to denote the random variable for the index j^*. Finally, for any $i \in [g_r]$ and $j \in [w_r]$, I_j^i denotes the random variable for the instance I_j^i. We start by stating a simple property of oblivious mapping functions.

Proposition 8.4.1. *For any $i \in [g_r]$ and any player $q \in P_i$, conditioned on input $(I^i(q), \phi_q)$ to player q, the index $j^* \in [w_r]$ is chosen uniformly at random.*

Proof. By Condition (2) of obliviousness in Definition 8.5, $\Phi_q \perp \mathsf{J}$, and hence $\mathsf{J} \perp \Phi_q = \phi_q$. Moreover, by Condition (1) of Definition 8.5, Φ_q cannot depend on $\mathsf{I}^i(q)$ and hence $\mathsf{I}^i(q) \perp \Phi_q = \phi_q$ also. Now notice that while the distribution of I_j^i and $\mathsf{I}_{j^*}^i$ for $j \neq j^*$, i.e., $\mathcal{D}'_{r-1}$ and $\mathcal{D}_{r-1}$ are different, the distribution of $\mathsf{I}_j^i(q)$ and $\mathsf{I}_{j^*}^i(q)$ are identical by definition of $\mathcal{D}'_{r-1}$. As such, $\mathsf{I}^i(q)$ and j^* are also independent of each other conditioned on $\Phi_q = \phi_q$, finalizing the proof. □

We show that any protocol with a small communication cost cannot learn essentially any useful information about the special instance $I_r^\star$ in its first round.

Lemma 8.4.2. *For any deterministic protocol π for $\mathcal{D}_r$, $\mathbb{I}(\mathsf{I}_r^\star \,; \Pi_1 \mid \Phi, \mathsf{J}) \leq |\Pi_1|/w_r$.*

Proof. The first step is to show that the information revealed about $\mathsf{I}_r^\star$ via Π_1 can be partitioned over the messages sent by each individual player about their own input in their special instance.

Claim 8.4.3. $\mathbb{I}(\mathsf{I}_r^\star \,; \Pi_1 \mid \Phi, \mathsf{J}) \leq \sum_{q \in P} \mathbb{I}(\mathsf{I}_r^\star(q) \,; \Pi_{1,q} \mid \Phi, \mathsf{J})$.

Proof. Intuitively, the claim is true because after conditioning on Φ and J, the input of players become independent of each other on all fooling instances, i.e., every instance except for their copy of $I_r^\star$. As a result, the messages communicated by one player do not add extra information to messages of another one about $I_r^\star$. Moreover, since each player q is observing $I_r^\star(q)$, the information revealed by this player can only be about $I_r^\star(q)$ and not $I_r^\star$. We now provide the formal proof. Recall that $\Pi_1 = (\Pi_{1,1}, \ldots, \Pi_{1,p_r})$. By chain rule of mutual

information,

$$\mathbb{I}(\mathsf{I}_r^\star\,;\Pi_1\mid\Phi,\mathsf{J})\underset{\text{Fact 2.6.1-(6)}}{=}\sum_{q\in P}\mathbb{I}(\mathsf{I}_r^\star\,;\Pi_{1,q}\mid\Pi_1^{<q},\Phi,\mathsf{J}).$$

We first show that for each $q\in P$,

$$\mathbb{I}(\mathsf{I}_r^\star\,;\Pi_{1,q}\mid\Pi_1^{<q},\Phi,J)\le\mathbb{I}(\mathsf{I}_r^\star\,;\Pi_{1,q}\mid\Phi,\mathsf{J}).\tag{8.1}$$

Recall that, for any player q, $\mathsf{I}(q)$ denotes the input to player q in all instances in which q is participating, and define $\mathsf{I}(-q)$ as the collection of the inputs to all other players across all instances. We argue that $\mathsf{I}(q)\perp\mathsf{I}(-q)\mid\mathsf{I}_r^\star,\Phi,\mathsf{J}$. The reason is simply because after conditioning on $\mathsf{I}_r^\star$, the only variables in $\mathsf{I}(q)$ and $\mathsf{I}(-q)$ are fooling instances that are sampled from $\mathcal{D}'^{r-1}$ which is a product distribution across players. This implies that $\mathbb{I}(\mathsf{I}(q)\,;\mathsf{I}(-q)\mid\mathsf{I}_r^\star,\Phi,\mathsf{J})=0$ (by Fact 2.6.1-(2)). Now, notice that the input to each player q is uniquely identified by $(\mathsf{I}(q),\Phi)$ (by locally computable property in Definition 8.3) and hence conditioned on $\mathsf{I}_r^\star,\Phi,J$, the message $\Pi_{1,q}$ is a deterministic function of $\mathsf{I}(q)$. As such, by the data processing inequality (Fact 2.6.1-(7)), we have that $\mathbb{I}(\Pi_{1,q}\,;\Pi_1^{<q}\mid\mathsf{I}_r^\star,\Phi,\mathsf{J})=0$; by Proposition 2.6.4, this implies Eq (8.1) (here, conditioning on $\Pi_1^{<q}$ in RHS of Eq (8.1) can only decrease the mutual information).

Define $\mathsf{I}_r^\star(-q)$ as the input to all players in $\mathsf{I}_r^\star$ except for player q; hence $\mathsf{I}_r^\star=(\mathsf{I}_r^\star(q),\mathsf{I}_r^\star(-q))$. By chain rule of mutual information (Fact 2.6.1-(6)),

$$\mathbb{I}(\mathsf{I}_r^\star\,;\Pi_{1,q}\mid\Phi,\mathsf{J})=\mathbb{I}(\mathsf{I}_r^\star(q)\,;\Pi_{1,q}\mid\Phi,\mathsf{J})+\mathbb{I}(\mathsf{I}_r^\star(-q)\,;\Pi_{1,q}\mid\mathsf{I}_r^\star(q),\Phi,\mathsf{J})=\mathbb{I}(\mathsf{I}_r^\star(q)\,;\Pi_{1,q}\mid\Phi,\mathsf{J})$$

since $\mathbb{I}(\mathsf{I}_r^\star(-q)\,;\Pi_{1,q}\mid\mathsf{I}_r^\star(q),\Phi,\mathsf{J})=0$ as $\Pi_{1,q}$ is independent of $\mathsf{I}_r^\star(-q)$ after conditioning on $\mathsf{I}_r^\star(q)$ (and Fact 2.6.1-(2)). The claim now follows from Eq (8.1) and above. □ Claim 8.4.3

Next, we use a direct-sum style argument to show that as each player is oblivious to the identity of the special instance in the input, the message sent by this player cannot reveal much information about the special instance, unless it is too large.

Claim 8.4.4. *For any group P_i and player $q\in P_i$, $\mathbb{I}(\mathsf{I}_r^\star(q)\,;\Pi_{1,q}\mid\Phi,\mathsf{J})\le|\Pi_{1,q}|/w_r$.*

Proof. We first argue that,

$$\mathbb{I}(\mathsf{I}_r^\star(q)\,;\Pi_{1,q}\mid\Phi,\mathsf{J})\le\mathbb{I}(\mathsf{I}_r^\star(q)\,;\Pi_{1,q}\mid\Phi_q,\mathsf{J}).\tag{8.2}$$

Let $\Phi=(\Phi_q,\Phi^{-q})$ where Φ^{-q} denotes the rest of the mapping function Φ beyond Φ_q. We have, $\Pi_{1,q}\perp\Phi^{-q}\mid\Phi_q,\mathsf{J},\mathsf{I}_r^\star(q)$ since after conditioning on J, Φ does not depend on any

other variable in $\mathcal{D}_r$ (by obliviousness property in Definition 8.5), and hence the input to player q and as a result $\Pi_{1,q}$ are independent of Φ^{-q} after conditioning on both Φ_q and J. Eq (8.2) now follows from the independence of $\Pi_{1,q}$ and Φ^{-q} and Proposition 2.6.4 (as conditioning on $\Phi^{<q}$ in RHS of Eq (8.2) can only decrease the mutual information).

We can bound the RHS of Eq (8.2) as follows,

$$\mathbb{I}(\mathsf{I}^\star_r(q)\ ;\Pi_{1,q}\mid\Phi_q,\mathsf{J})=\mathop{\mathbb{E}}_{j\in[w_r]}\left[\mathbb{I}(\mathsf{I}^\star_r(q)\ ;\Pi_{1,q}\mid\Phi_q,\mathsf{J}=j)\right]=\frac{1}{w_r}\sum_{j=1}^{w_r}\mathbb{I}(\mathsf{I}^i_j(q)\ ;\Pi_{1,q}\mid\Phi_q,\mathsf{J}=j).$$

(j^* is chosen uniformly at random from $[w_r]$ and $\mathsf{I}^\star_r=\mathsf{I}^i_j$ conditioned on $\mathsf{J}=j$)

Our goal now is to drop the conditioning on the event $\mathsf{J}=j$. By Definition 8.5, Φ_q is independent of $\mathsf{J}=j$. Moreover, $\mathsf{I}^i_j(q)$ is sampled from $\mathcal{D}_{r-1}(q)$ (both in $\mathcal{D}_{r-1}$ and in $\mathcal{D}'_{r-1}$) and hence is independent of $\mathsf{J}=j$, even conditioned on Φ_q. Finally, by Proposition 8.4.1, the input to player q is independent of $\mathsf{J}=j$ and as $\Pi_{1,q}$ is a deterministic function of the input to player q, $\Pi_{1,q}$ is also independent of $\mathsf{J}=j$, even conditioned on Φ_q and $\mathsf{I}^i_j(q)$. This means that the joint distribution of $\mathsf{I}^i_j(q),\Pi_{1,q}$, and Φ_q is independent of the event $\mathsf{J}=j$ and hence we can drop this conditioning in the above term, and obtain that,

$$\begin{aligned}\frac{1}{w_r}\sum_{j=1}^{w_r}\mathbb{I}(\mathsf{I}^i_j(q)\ ;\Pi_{1,q}\mid\Phi_i,\mathsf{J}=j)&=\frac{1}{w_r}\sum_{j=1}^{w_r}\mathbb{I}(\mathsf{I}^i_j(q)\ ;\Pi_{1,q}\mid\Phi_i)\\&\le\frac{1}{w_r}\sum_{j=1}^{w_r}\mathbb{I}(\mathsf{I}^i_j(q)\ ;\Pi_{1,q}\mid\mathsf{I}^{i,<j}(q),\Phi_i)=\frac{1}{w_r}\cdot\mathbb{I}(\mathsf{I}^i(q)\ ;\Pi_{1,q}\mid\Phi_i),\end{aligned}$$

where the inequality holds since $\mathsf{I}^i_j(q)\perp\mathsf{I}^{i,<j}(q)\mid\Phi_i$ and hence conditioning on $\mathsf{I}^{i,<j}(q)$ can only increase the mutual information by Proposition 2.6.3. Finally,

$$\begin{aligned}\frac{1}{w_r}\cdot\mathbb{I}(\mathsf{I}^i(q)\ ;\Pi_{1,q}\mid\Phi_i)&\underset{\text{Fact 2.6.1-(1)}}{\le}\frac{1}{w_1}\cdot\mathcal{H}(\Pi_{1,q}\mid\Phi_i)\\&\underset{\text{Fact 2.6.1-(3)}}{\le}\frac{1}{w_r}\cdot\mathcal{H}(\Pi_{1,q})\underset{\text{Fact 2.6.1-(1)}}{\le}\frac{1}{w_r}\cdot|\Pi_{1,q}|\,,\end{aligned}$$

finalizing the proof. □ Claim 8.4.4

Lemma 8.4.2 now follows from the previous two claims:

$$\mathbb{I}(\mathsf{I}^\star_r\ ;\Pi_1\mid\Phi,\mathsf{J})\underset{\text{Claim 8.4.3}}{\le}\sum_{q\in P}\mathbb{I}(\mathsf{I}^\star_r(q)\ ;\Pi_{1,q}\mid\Phi,\mathsf{J})\underset{\text{Claim 8.4.4}}{\le}\frac{1}{w_r}\cdot\sum_{q\in P}|\Pi_{1,q}|=\frac{1}{w_r}\cdot|\Pi_1|\,.$$

This concludes the proof. □ Lemma 8.4.2

For any tuple (Π_1, ϕ, j), we define the distribution $\psi(\Pi_1, \phi, j)$ as the distribution of $\mathsf{I}_r^\star$ in $\mathcal{D}_r$ conditioned on $\mathsf{\Pi}_1 = \Pi_1$, $\mathsf{\Phi} = \phi$, and $\mathsf{J} = j$. Recall that the original distribution of $\mathsf{I}_r^\star$ is $\mathcal{D}_{r-1}$. In the following, we show that if the first message sent by the players is not too large, and hence does not reveal much information by about $I_r^\star$ by Lemma 8.4.2, even after the aforementioned conditioning, distribution of $\mathsf{I}_r^\star$ does not change by much in average.

Lemma 8.4.5. *If $|\mathsf{\Pi}_1| = o(w_r/r^4)$, then $\mathbb{E}_{(\Pi_1,\phi,j)}\left[\,|\psi(\Pi_1,\phi,j) - \mathcal{D}_{r-1}|_{tvd}\,\right] = o(1/r^2)$.*

Proof. Since $I_r^\star$ is independent of ϕ and j^* in $\mathcal{D}_r$, we have $\mathcal{D}_{r-1} = \textsc{dist}(\mathsf{I}_r^\star) = \textsc{dist}(\mathsf{I}_r^\star \mid \mathsf{\Phi}, \mathsf{J})$. As such, it suffices to show that $\textsc{dist}(\mathsf{I}_r^\star \mid \mathsf{\Phi}, \mathsf{J})$ is close to the distribution of $\textsc{dist}(\mathsf{I}_r^\star \mid \mathsf{\Pi}_1, \mathsf{\Phi}, \mathsf{J})$. By Lemma 8.4.2 and the assumption $|\mathsf{\Pi}_1| = o(w_r/r^4)$, we know that the information revealed about $I_r^\star$ by $\mathsf{\Pi}_1$, conditioned on $\mathsf{\Phi}, \mathsf{J}$ is quite small, i.e., $o(1/r^4)$. This intuitively means that having an extra knowledge of $\mathsf{\Pi}_1$ would not be able to change the distribution of $I_r^\star$ by much. We now formalizes this intuition.

$$
\begin{aligned}
\mathop{\mathbb{E}}_{(\Pi_1,\phi,j)}\left[\,|\psi(\Pi_1,\phi,j) - \mathcal{D}_{r-1}|_{tvd}\,\right] &= \mathop{\mathbb{E}}_{(\Pi_1,\phi,j)}\left[\,|\textsc{dist}(\mathsf{I}_r^\star \mid \Pi_1,\phi,j) - \textsc{dist}(\mathsf{I}_r^\star \mid \phi,j)|_{tvd}\,\right] \\
&\le \mathop{\mathbb{E}}_{(\Pi_1,\phi,j)}\left[\sqrt{\frac{1}{2}\cdot\mathbb{D}(\textsc{dist}(\mathsf{I}_r^\star \mid \Pi_1,\phi,j) \;||\; \textsc{dist}(\mathsf{I}_r^\star \mid \phi,j))}\right] \\
&\qquad \text{(By Pinsker's inequality (Fact 2.6.8); here } \mathbb{D} \text{ is the KL-divergence)} \\
&\le \sqrt{\frac{1}{2}\cdot\mathop{\mathbb{E}}_{(\Pi_1,\phi,j)}\left[\mathbb{D}(\textsc{dist}(\mathsf{I}_r^\star \mid \Pi_1,\phi,j) \;||\; \textsc{dist}(\mathsf{I}_r^\star \mid \phi,j))\right]} \\
&\qquad \text{(By concavity of } \sqrt{\cdot} \text{ and Jensen's inequality)} \\
&\underset{\text{Fact 2.6.6}}{=} \sqrt{\frac{1}{2}\cdot\mathbb{I}(\mathsf{I}_r^\star\,;\mathsf{\Pi}_1 \mid \mathsf{\Phi},\mathsf{J})} \underset{\text{Lemma 8.4.2}}{\le} \sqrt{\frac{1}{2}\cdot\frac{1}{w_r}\cdot|\mathsf{\Pi}_1|},
\end{aligned}
$$

which is $o(1/r^2)$ as $|\mathsf{\Pi}_1| = o(w_r/r^4)$. □ Lemma 8.4.5

Define the recursive function $\delta(r) := \delta(r-1) - o(1/r^2) - \gamma$ with base $\delta(0) = 1/2$. We have,

Lemma 8.4.6. *For any deterministic $\delta(r)$-error r-round protocol π for $\mathcal{D}_r$, $\|\pi\| = \Omega(w_r/r^4)$.*

Proof. The proof is by induction on the number of rounds r.

Base case: The base case of this lemma refers to 0-round protocols for $\mathcal{D}_0$, i.e., protocols that are not allowed any communication. As in the distribution $\mathcal{D}_0$, Yes and No instances happen w.p. $1/2$ each and the coordinator has no input, any 0-round protocol can only output the correct answer w.p. $1/2$, proving the induction base.

Induction step: Suppose the lemma holds for all integers up to r and we prove it for r round protocols. The proof is by contradiction. Given an r-round protocol π_r violating the induction hypothesis, we create an $(r-1)$-round protocol π_{r-1} which also violates the

induction hypothesis, a contradiction. Given an instance I_{r-1} of $\mathcal{P}_{s_{r-1}}$ over players P^{r-1} and domain $D^{r-1} = \{0,1\}^{s_{r-1}}$, the protocol π_{r-1} works as follows:

1. Let $P^r = [p_r]$ and partition P^r into g_r equal-size groups $P_1, \ldots, P_{g_r}$ as is done in $\mathcal{D}_r$. Create an instance I_r of $\mathcal{D}_r$ as follows:
2. Using *public randomness*, the players in P^{r-1} sample $R := (\Pi_1, \phi, j^*) \sim (\textsc{dist}(\pi_r), \mathcal{D}_r)$, from the (joint) distribution of protocol π_r over distribution $\mathcal{D}_r$.
3. The q-th player in P^{r-1} (in instance I_{r-1}) mimics the role of the q-th player in each group P_i for $i \in [g_r]$ in I_r, denoted by player (i,q), as follows:
   (a) Set the input for (i,q) in the special instance $I^i_{j^*}(q)$ of I_r as the original input of q in I_{r-1}, i.e., $I_{r-1}(q)$ mapped via σ_r and ϕ to I (as is done in I_r to the domain $D^i_{j^*}$). This is possible by the locally computable property of σ_r and ϕ in Definition 8.3.
   (b) Sample the input for (i,q) in all the fooling instances $I^i_j(q)$ of I_r for any $j \neq j^*$ using *private randomness* from the *correlated* distribution $\mathcal{D}_r \mid (\mathsf{I}^\star_r = I_{r-1}, (\Pi_1, \Phi, \mathsf{J}) = R)$. This sampling is possible by Proposition 8.4.7.
4. Run the protocol π_r from the second round onwards on I_r assuming that in the first round the communicated message was Π_1 and output the same answer as π_r.

Notice that in Line (3b), the distribution the players are sampling from depends on Π_1, ϕ, j^* which are public knowledge (through sampling via public randomness), as well as $I^\star_r$ which is *not* a public information as each player q only knows $I^\star_r(q)$ and not all of $I^\star_r$. Moreover, while random variables $\mathsf{I}^i_j(q)$ (for $j \neq j^*$) are originally independent across different players q (as they are sampled from the product distribution $\mathcal{D}'_{r-1}$), conditioning on the first message of the protocol, i.e., Π_1 correlates them, and hence a-priori it is not clear whether the sampling in Line (3b) can be done without any further communication. Nevertheless, we can prove that this is the case and to sample from the distribution in Line (3b), each player only needs to know $I^\star_r(q)$ and not $I^\star_r$.

Proposition 8.4.7. *Suppose* I *is the collection of all instances in the distribution* $\mathcal{D}_r$ *and* $\mathsf{I}(q)$ *is the input to player* q *in instances in which* q *participates; then,*

$$\textsc{dist}(\mathsf{I} \mid \mathsf{I}^\star_r = I_{r-1}, (\Pi_1, \Phi, \mathsf{J}) = R) = \mathsf{X}_{q \in P}\ \textsc{dist}(\mathsf{I}(q) \mid \mathsf{I}^\star_r(q) = I_{r-1}(q), (\Pi_1, \Phi, \mathsf{J}) = R).$$

Proof. Fix any player $q \in P$, and recall that $\mathsf{I}(-q)$ is the collection of the inputs to all players other than q across all instances (special and fooling). We prove that $\mathsf{I}(q) \perp \mathsf{I}(-q) \mid (\mathsf{I}^\star_r(q), \Pi_1, \Phi, \mathsf{J})$ in $\mathcal{D}_r$, which immediately implies the result. To prove this claim, by Fact 2.6.1-(2), it suffices to show that $\mathbb{I}(\mathsf{I}(q)\ ; \mathsf{I}(-q) \mid \mathsf{I}^\star_r(q), \Pi_1, \Phi, \mathsf{J}) = 0$. Define Π_1^{-q} as

the set of all messages in Π_1 except for the message of player q, i.e., $\Pi_{1,q}$. We have,

$$\mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Pi_1,\Phi,\mathsf{J}) \le \mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Pi_{1,q},\Phi,\mathsf{J}),$$

since $\mathsf{I}(q) \perp \Pi_1^{-q} \mid \mathsf{I}(-q),\mathsf{I}^\star_r(q),\Pi_{1,q},\Phi,\mathsf{J}$ as the input to players $P\setminus\{q\}$ is uniquely determined by $\mathsf{I}(-q),\Phi$ (by the locally computable property in Definition 8.3) and hence Π_1^{-q} is deterministic after the conditioning; this independence means that conditioning on Π_1^{-q} in the RHS above can only decrease the mutual information by Proposition 2.6.4. We can further bound the RHS above by,

$$\mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Pi_{1,q},\Phi,\mathsf{J}) \le \mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Phi,\mathsf{J}),$$

since $\mathsf{I}(-q) \perp \Pi_{1,q} \mid \mathsf{I}(q),\mathsf{I}^\star_r(q),\Phi,\mathsf{J}$ as the input to player q is uniquely determined by $\mathsf{I}(q),\Phi$ (again by Definition 8.3) and hence after the conditioning, $\Pi_{1,q}$ is deterministic; this implies that conditioning on $\Pi_{1,q}$ in RHS above can only decrease the mutual information by Proposition 2.6.4. Finally, observe that $\mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Phi,\mathsf{J}) = 0$ by Fact 2.6.1-(2), since after conditioning on $I^\star_r(q)$, the only remaining instances in $\mathsf{I}(q)$ are fooling instances which are sampled from the distribution $\mathcal{D}'_{r-1}$ which is independent across the players. This implies that $\mathbb{I}(\mathsf{I}(q)\ ;\mathsf{I}(-q)\mid \mathsf{I}^\star_r(q),\Pi_1,\Phi,\mathsf{J}) = 0$ also which finalizes the proof. □ Proposition 8.4.7

Having proved Proposition 8.4.7, it is now easy to see that π_{r-1} is indeed a valid $r-1$ round protocol for distribution $\mathcal{D}_{r-1}$: each player q can perform the sampling in Line (3b) without any communication as $(I^\star(q),\Pi_1,\Phi,J)$ are all known to q; this allows the players to simulate the first round of protocol π_r without any communication and hence only need $r-1$ rounds of communication to compute the answer of π_r. We can now prove that,

Claim 8.4.8. *Assuming π_r is a δ-error protocol for $\mathcal{D}_r$, π_{r-1} would be a $\big(\delta+\gamma+o(1/r^2)\big)$-error protocol for $\mathcal{D}_{r-1}$.*

Proof. Our goal is to calculate the probability that π_{r-1} errs on an instance $I_{r-1}\sim\mathcal{D}_{r-1}$. For the sake of analysis, suppose that I_{r-1} is instead sampled from the distribution ψ for a randomly chosen tuple (Π_1,ϕ,j^*) (defined before Lemma 8.4.5). Notice that by Lemma 8.4.5, these two distributions are quite close to each other in total variation distance, and hence if π_{r-1} has a small error on distribution ψ it would necessarily has a small error on $\mathcal{D}_{r-1}$ as well (by Fact 2.6.7).

Using Proposition 8.4.7, it is easy to verify that if I_{r-1} is sampled from ψ, then the instance I_r constructed by π_{r-1} is sampled from $\mathcal{D}_r$ and moreover $I^\star_r = I_{r-1}$. As such, since (i) π_r is a δ-error protocol for $\mathcal{D}_r$, (ii) the answer to I_r and $I^\star_r = I_{r-1}$ are the same w.p. $1-\gamma$ (by γ-preserving property in Definition 8.4), and (iii) π_{r-1} outputs the same answer

as π_r, protocol π_{r-1} is a $(\delta+\gamma)$-error protocol for ψ. We now prove this claim formally. Define $\mathsf{R}^{\mathsf{pri}}$ and $\mathsf{R}^{\mathsf{pub}}$ as, respectively, the private and public randomness used by π_{r-1}.

$$\begin{aligned}
\Pr_{\mathcal{D}_{r-1}}(\pi_{r-1} \text{ errs}) &= \mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\mathcal{D}_{r-1}} \mathop{\mathbb{E}}_{\mathsf{R}^{\mathsf{pub}}} \left[\Pr_{\mathsf{R}^{\mathsf{pri}}} \left(\pi_{r-1} \text{ errs} \mid \mathsf{R}^{\mathsf{pub}}\right) \right] \\
&= \mathop{\mathbb{E}}_{(\Pi_1,\phi,j^*)} \mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\mathcal{D}_{r-1}|(\Pi_1,\phi,j^*)} \left[\Pr_{\mathsf{R}^{\mathsf{pri}}} (\pi_{r-1} \text{ errs} \mid \Pi_1,\phi,j^*) \right] \\
&\qquad\qquad (\text{as } \mathsf{R}^{\mathsf{pub}} \perp \mathsf{I}_{r-1} \text{ and } \mathsf{R}^{\mathsf{pub}} = (\Pi_1,\psi,j^*) \text{ in protocol } \pi_{r-1}) \\
&\le \mathop{\mathbb{E}}_{(\Pi_1,\phi,j^*)} \left[\mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\psi(\Pi_1,\phi,j^*)} \left[\Pr_{\mathsf{R}^{\mathsf{pri}}} (\pi_{r-1} \text{ errs} \mid \Pi_1,\phi,j^*) \right] + |\mathcal{D}_{r-1} - \psi(\Pi_1,\phi,j^*)|_{tvd} \right] \\
&\qquad\qquad (\text{by Fact 2.6.7 for distributions } \mathcal{D}_{r-1} \text{ and } \psi(\Pi_1,\phi,j^*)) \\
&= \mathop{\mathbb{E}}_{(\Pi_1,\phi,j^*)} \mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\psi(\Pi_1,\phi,j^*)} \left[\Pr_{\mathsf{R}^{\mathsf{pri}}} (\pi_{r-1} \text{ errs} \mid \Pi_1,\phi,j^*) \right] + o(1/r^2) \\
&\qquad\qquad (\text{by linearity of expectation and Lemma 8.4.5}) \\
&= \mathop{\mathbb{E}}_{(\Pi_1,\phi,j^*)} \mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\psi(\Pi_1,\phi,j^*)} \left[\Pr_{\mathcal{D}_r} (\pi_{r-1} \text{ errs} \mid \mathsf{I}^\star_r = I_{r-1},\Pi_1,\phi,j^*) \right] + o(1/r^2) \\
&\qquad\qquad (\textsc{dist}(\mathsf{R}^{\mathsf{pri}}) = \mathcal{D}_r \mid \mathsf{I}^\star_r = I_{r-1},\Pi_1,\phi,j^*) \\
&\le \mathop{\mathbb{E}}_{(\Pi_1,\phi,j^*)} \mathop{\mathbb{E}}_{\mathsf{I}_{r-1}\sim\psi(\Pi_1,\phi,j^*)} \left[\Pr_{\mathcal{D}_r} (\pi_r \text{ errs} \mid \mathsf{I}^\star_r = I_{r-1},\Pi_1,\phi,j^*) \right] + \gamma + o(1/r^2) \\
&(\mathcal{P}_{s_r}(I_r) = \mathcal{P}_{s_{r-1}}(I_{r-1}) \text{ w.p. } 1-\gamma \text{ by Definition 8.4 and } \pi_{r-1} \text{ outputs the same as } \pi_r) \\
&= \mathop{\mathbb{E}}_{(I^\star_r,\Pi_1,\phi,j^*)\sim\mathcal{D}_r} \left[\Pr_{\mathcal{D}_r} (\pi_r \text{ errs} \mid \mathsf{I}^\star_r = I_{r-1},\Pi_1,\phi,j^*) \right] + \gamma + o(1/r^2) \\
&\qquad\qquad (\psi(\Pi_1,\phi,j^*) = \textsc{dist}(\mathsf{I}^\star_r \mid \Pi_1,\phi,j^*) \text{ in } \mathcal{D}_r \text{ by definition}) \\
&= \Pr_{\mathcal{D}_r} (\pi_r \text{ errs}) + o(1/r^2) \le \delta+\gamma+o(1/r^2), \\
&(\text{as } \pi_r \text{ is a } \delta_r\text{-error protocol for } \mathcal{D}_r \text{ by the assumption in the lemma statement})
\end{aligned}$$

finalizing the proof. □ Claim 8.4.8

We are now ready to finalize the proof of Lemma 8.4.6. Suppose π_r is a deterministic $\delta(r)$-error protocol for $\mathcal{D}_r$ with communication cost $\|\pi_r\| = o(w_r/r^4)$. By Claim 8.4.8, π_{r-1} would be a randomized $\delta(r-1)$-error protocol for $\mathcal{D}_{r-1}$ with $\|\pi_{r-1}\| \le \|\pi_r\|$ (as $\delta(r-1) = \delta(r) + \gamma + o(1/r^2)$). By an averaging argument, we can fix the randomness in π_{r-1} to obtain a deterministic protocol π'_{r-1} over the distribution $\mathcal{D}_{r-1}$ with the same error $\delta(r-1)$ and communication of $\|\pi'_{r-1}\| = o(w_r/r^4) = o(w_{r-1}/r^4)$ (as $\{w_r\}_{r\ge 0}$ is a non-increasing sequence). But such a protocol contradicts the induction hypothesis for $(r-1)$-round protocols, finalizing the proof. □ Lemma 8.4.6

Proof of Theorem 8.3. By Lemma 8.4.6, any deterministic $\delta(r)$-error r-round protocol for $\mathcal{D}_r$ requires $\Omega(w_r/r^4)$ total communication. This immediately extends to randomized protocols by an averaging argument, i.e., the easy direction of Yao's minimax principle (Propo-

sition 2.7.2). The statement in the theorem now follows from this since for any $r \geq 0$, $\delta(r) = \delta(r-1) - \gamma - o(1/r^2) = \delta(0) - r \cdot \gamma - \sum_{\ell=1}^{r} o(1/\ell^2) = 1/2 - r \cdot \gamma - o(1) > 1/3 - r \cdot \gamma$ (as $\delta(0) = 1/2$ and $\sum_{\ell=1}^{r} 1/\ell^2$ is a converging series and hence is bounded by some absolute constant independent of r). □

8.5. A Distributed Lower Bound for Maximum Coverage

We prove our main lower bound for maximum coverage, formalizing Result 8.1.

Theorem 8.4. *For integers $1 \leq r, c \leq o\left(\frac{\log k}{\log\log k}\right)$ with $c \geq 4r$, any r-round protocol for the maximum coverage problem that can approximate the value of optimal solution to within a factor of better than $\left(\frac{1}{2c} \cdot \frac{k^{1/2r}}{\log k}\right)$ w.p. at least $3/4$ requires $\Omega\left(\frac{k}{r^4} \cdot m^{\frac{c}{(c+2)\cdot 4r}}\right)$ communication per machine. The lower bound applies to instances with m sets, $n = m^{1/\Theta(c)}$ elements, and $k = \Theta(n^{2r/(2r+1)})$.*

The proof is based on an application of Theorem 8.3. In the following, let $c \geq 1$ be any integer (as in Theorem 8.4) and $N \geq 12c^2$ be a sufficiently large integer which we use to define the main parameters for our problem. To invoke Theorem 8.3, we need to instantiate the recursive family of distributions $\{\mathcal{D}_r\}_{r=0}^{c}$ in Section 8.4 with appropriate sequences and gadgets for the maximum coverage problem. We first define sequences (for all $0 \leq r \leq c$):

$$k_r = p_r = (N^2 - N)^r, \quad n_r = N^{2r+1}, \quad m_r = \left(N^c \cdot (N^2 - N)\right)^r, \quad w_r = N^c \quad g_r = (N^2 - N)$$

Here, m_r, n_r, and k_r, respectively represent the number of sets and elements and the parameter k in the maximum coverage problem in the instances of each distribution $\mathcal{D}_r$ and together can identify the size of each instance (i.e., the parameter s_r defined in Section 8.4 for the distribution $\mathcal{D}_r$). Moreover, p_r, w_r and g_r represent the number of players, the width parameter, and the number of groups in $\mathcal{D}_r$, respectively (notice that $g_r = p_r/p_{r-1}$ as needed in distribution $\mathcal{D}_r$). Using the sequences above, we define:

coverage(N, r): the problem of deciding whether the optimal k_r cover of universe $[n_r]$ with m_r input sets is at least $(k_r \cdot N)$ (Yes case), or at most $\left(k_r \cdot 2c \cdot \log\left(N^{2r}\right)\right)$ (No case).

There is a gap of roughly $N \approx k_r^{1/2r}$ (ignoring the lower order terms) between the optimal solution in Yes and No cases of coverage(N, r). We prove a lower bound for deciding between Yes and No instances of coverage(N, r), when the input sets are partitioned between the players, which implies an identical lower bound for algorithms that can approximate the value of optimal solution in maximum coverage to within a factor smaller than $k_r^{1/2r}$.

Recall that to use the framework introduced in Section 8.4, one needs to define two problem-specific gadgets, i.e., a packing function, and a labeling family. In the following section, we design a crucial building block for our packing function.

RND Set-Systems. Our packing function is based on the following set-system.

Definition 8.6. *For integers $N, r, c \geq 1$, an (N, r, c)-*randomly nearly disjoint (RND) *set-system over a universe $\mathcal{X}$ of N^{2r} elements, is a collection $\mathcal{S}$ of subsets of $\mathcal{X}$ satisfying the following properties:*

1. *Each set $A \in \mathcal{S}$ is of size N^{2r-1}.*
2. *Fix any set $B \in \mathcal{S}$ and suppose $\mathcal{C}_B$ is a collection of $N^{c \cdot r}$ subsets of $\mathcal{X}$ whereby each set in $\mathcal{C}_B$ is chosen by picking an arbitrary set $A \neq B$ in $\mathcal{S}$, and then picking an N-subset uniformly at random from A (we do* not *assume independence between the sets in $\mathcal{C}_B$). Then,*

$$\Pr\left(\exists\ S \in \mathcal{C}_B \ \ s.t.\ \ |S \cap B| \geq 2c \cdot r \cdot \log N\right) = o(1/N^3).$$

Intuitively, this means that any random N-subset of some set $A \in \mathcal{S}$ is essentially disjoint from any other set $B \in \mathcal{S}$ w.h.p.

Existence of large RND set-systems follows from probabilistic method (see our paper [31]).

Lemma 8.5.1 ([31])**.** *For integers $1 \leq r \leq c$ and sufficiently large integer $N \geq c$, there exists an (N, r, c)-RND set-system $\mathcal{S}$ of size N^c over any universe $\mathcal{X}$ of size N^{2r}.*

8.5.1. Proof of Theorem 8.4

We parameterize the recursive family of distributions $\{\mathcal{D}_r\}_{r=0}^{c}$ in our framework in Section 8.4 for the coverage problem, i.e., $\mathsf{coverage}(N, r,)$, with the aforementioned sequences plus the packing and labeling functions which we define below.

Packing function σ_r: Mapping instances $I^i_1, \ldots, I^i_{w_r}$ each over $n_{r-1} = N^{2r-1}$ elements and m_{r-1} sets for any group $i \in [g_r]$ to a single instance I^i on N^{2r} elements and $w_r \cdot m_{r-1}$ sets.

1. Let $\mathcal{A} = \{A_1, \ldots, A_{w_r}\}$ be an (N, r, c)-RND system with $w_r = N^c$ sets over some universe $\mathcal{X}_i$ of N^{2r} elements (guaranteed to exist by Lemma 8.5.1 since $c < N$). By definition of $\mathcal{A}$, for any set $A_j \in \mathcal{A}$, $|A_j| = N^{2r-1} = n_{r-1}$.
2. Return the instance I over the universe $\mathcal{X}_i$ with the collection of all sets in $I^i_1, \ldots, I^i_{w_r}$ after mapping the elements in I^i_j to A_j arbitrarily.

We now define the labeling family Φ_r as a function of the index $j^* \in [w_r]$ of special instances.

Labeling family Φ_r: Mapping instances $I^1, \ldots, I^{g_r}$ over N^{2r} elements to a single instance I on $n_r = N^{2r+1}$ elements and m_r sets.

1. Let $j^* \in [w_r]$ be the index of the special instance in the distribution $\mathcal{D}_r$. For each permutation π of $[N^{2r+1}]$ we have a unique function $\phi(j^*, \pi)$ in the family.
2. For any instance I^i for $i \in [g_r]$, map the elements in $\mathcal{X}_i \setminus A_{j^*}$ to $\pi(1, \ldots, N^{2r} - N^{2r-1})$ and the elements in A_{j^*} to $\pi(N^{2r} + (g_r - 1) \cdot N^{2r-1}) \ldots \pi(N^{2r} + g_r \cdot N^{2r-1} - 1)$.
3. Return the instance I over the universe $[N^{2r+1}]$ which consists of the collection of all sets in $I^1, \ldots, I^{g_r}$ after the mapping above.

We define the base case distribution $\mathcal{D}_0$ of the recursive family $\{\mathcal{D}_r\}_{r=0}^{c}$. By definition of our sequences, this distribution is over $p_0 = 1$ player, $n_0 = N$ elements, and $m_0 = 1$ set.

Distribution $\mathcal{D}_0$: The base case of the recursive family of distributions $\{\mathcal{D}_r\}_{r=0}^{c}$.

1. W.p. $1/2$, the player has a single set of size N covering the universe (the Yes case).
2. W.p. $1/2$, the player has a single set $\{\emptyset\}$, i.e., a set that covers no elements (the No case).

To invoke Theorem 8.3, we prove that this family is a γ-hard recursive family for the parameter $\gamma = o(r/N)$. The sequences clearly satisfy the required monotonicity properties. It is also straightforward to verify that σ_r and functions $\phi \in \Phi_r$ are *locally computable* (Definition 8.3): both functions are specifying a mapping of elements to the new instance and hence each player can compute its final input by simply mapping the original input sets according to σ_r and ϕ to the new universe. In other words, the local mapping of each player $q \in P_i$ only specifies which element in the instance I corresponds to which element in $I^i_j(q)$ for $j \in [w_r]$. It thus remains to prove the *preserving* and *obliviousness* property of the packing and labeling functions.

We start by showing that the labeling family Φ_r is oblivious. The first property of Definition 8.5 is immediate to see as Φ_r is only a function of j^* and σ_r. For the second property, consider any group P_i and instance I^i; the labeling function never maps two elements belonging to a *single* instance I^i to the same element in the final instance (there are however overlaps between the elements across different groups). Moreover, picking a uniformly at random labeling function ϕ from Φ_r (as is done is $\mathcal{D}_r$) results in mapping the elements in I^i according to a *random permutation*; as such, the set of elements in instance I^i is mapped to a uniformly at random chosen subset of the elements in I, *independent* of the choice of j^*. As the local mapping ϕ_q of each player $q \in P_i$ is only a function of the set

of elements to which elements in I^i are mapped to, ϕ_q is also independent of j^*, proving that Φ_r is indeed oblivious.

The rest of this section is devoted to the proof of the preserving property of the packing and labeling functions defined for maximum coverage. We first make some observations about the instances created in $\mathcal{D}_r$. Recall that the special instances in the distribution are $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$. After applying the packing function, each instance $I^i_{j^*}$ is supported on the set of elements A_{j^*}. After additionally applying the labeling function, A_{j^*} is mapped to a unique set of elements in I (according to the underlying permutation π in ϕ); as a result,

Observation 8.5.2. *The elements in the special instances $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ are mapped to* disjoint *set of elements in the final instance.*

The input to each player $q \in P_i$ in an instance of $\mathcal{D}_r$ is created by mapping the sets in instances $I^i_1, \ldots, I^i_{w_r}$ (which are all sampled from distributions $\mathcal{D}_{r-1}$ or $\mathcal{D}'_{r-1}$) to the final instance I. As the packing and labeling functions, by construction, never map two elements belonging to the same instance I^i_j to the same element in the final instance, the size of each set in the input to player q is equal across any two distributions $\mathcal{D}_r$ and $\mathcal{D}_{r'}$ for $r \neq r'$, and thus is N by definition of $\mathcal{D}_0$ (we ignore empty sets in $\mathcal{D}_0$ as one can consider them as not giving any set to the player instead; these sets are only added to simplify that math). Moreover, as argued earlier, the elements are being mapped to the final instance according to a random permutation and hence,

Observation 8.5.3. *For any group P_i, any player $q \in P_i$, the distribution of any* single *input set to player q in the final instance $I \sim \mathcal{D}_r$ is* uniform *over all N-subsets of the universe. This also holds for an instance $I \sim \mathcal{D}'_r$ as marginal distribution of a player input is identical.*

We now prove the preserving property in the following two lemmas.

Lemma 8.5.4. *For any instance $I \sim \mathcal{D}_r$; if $I^\star_r$ is a* Yes *instance, then I is a* Yes *instance.*

Proof. Recall that the distribution of the special instance $I^\star_r$ is $\mathcal{D}_{r-1}$. Since $I^\star_r$ is a Yes instance, all $I^i_{j^*}$ for $i \in [g_r]$ are also Yes instances. By definition of $\mathsf{coverage}(N, r-1)$ and choice of k_{r-1}, this means that $\mathrm{opt}(I^i_{j^*}) \geq k_{r-1} \cdot N$. Moreover, by Observation 8.5.2, all copies of the special instance $I^\star_r$, i.e., $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ are supported on disjoint set of elements in I. As $k_r = k_{r-1} \cdot g_r$, we can pick the optimal solution from each $I^i_{j^*}$ for $i \in [g_r]$ and cover at least $k_r \cdot N$ elements. By definition of $\mathsf{coverage}(N, r)$, this implies that I is also a Yes instance. □

Lemma 8.5.5. *For any instance $I \sim \mathcal{D}_r$; if $I^\star_r$ is a* No *instance, then w.p. at least $1 - 1/N$,*

I is also a No *instance.*

Proof. Let U be the universe of elements in I and $U^\star \subseteq U$ be the set of elements to which the elements in special instances $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ are mapped to (these are all elements in U except for the first N^{2r} elements according to the permutation π in the labeling function ϕ). In the following, we bound the contribution of each set in players inputs in covering $U^\star$ and then use the fact that $|U \setminus U^\star|$ is rather small to finalize the proof.

For any group P_i for $i \in [g_r]$, let U_i be the set of all elements across instances in which the players in P_i are participating in. Moreover, define $U^\star_i := U^\star \cap U_i$; notice that $U^\star_i$ is precisely the set of elements in the special instance $I^i_{j^*}$. We first bound the contribution of special instances.

Claim 8.5.6. *If $I^\star_r$ is a* No *instance, then for any integer $\ell \geq 0$, any ℓ sets from the* special instances $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ *can cover at most $k_r + \ell \cdot (2c \cdot \log N^{2r-2})$ elements in $U^\star$.*

Proof. By definition of $\mathsf{coverage}(N, r-1)$, since $I^\star_r$ is a No instance, we have $\mathrm{opt}(I^\star_r) \leq k_{r-1} \cdot 2c \cdot \log(N^{2r-2})$. This implies that any collection of $\ell \geq k_{r-1}$ sets from $I^\star_r$ can only cover only $\ell \cdot 2c \cdot \log(N^{2r-2})$ elements; otherwise, by picking the best k_{r-1} sets among this collection, we can cover more that $\mathrm{opt}(I^\star_r)$, a contradiction. Now notice that since $I^\star_r$ is a No instance, we know that all instances $I^1_{j^*}, \ldots, I^{g_r}_{j^*}$ are also No instances. Thus, any collection of $\ell \geq k_{r-1}$ sets from each $I^i_{j^*}$ can cover at most $\ell \cdot 2c \cdot \log(N^{2r-2})$ elements in $U^\star$.

Let $\mathcal{C}$ be any collection of ℓ sets from special instances and $\mathcal{C}_i$ be the sets in $\mathcal{C}$ that are chosen from the instance $I^i_{j^*}$. Finally, let $\ell_i = |\mathcal{C}_i|$. We have (recall that $c(\mathcal{C})$ denotes the set of covered elements by $\mathcal{C}$),

$$\begin{aligned} |c(\mathcal{C}) \cap U^\star| &= \sum_{i \in [g_r]} |c(\mathcal{C}_i) \cap U^\star_i| \leq \sum_{i \in [g_r]} (k_{r-1} + \ell_i) \cdot 2c \cdot \log(N^{2r-2}) \\ &= g_r \cdot k_{r-1} + \ell \cdot 2c \cdot \log(N^{2r-2}) \leq k_r + \ell \cdot 2c \cdot \log(N^{2r-2}), \end{aligned}$$

where the last inequality holds because $g_r \cdot k_{r-1} = k_r$. □ Claim 8.5.6

We now bound the contribution of fooling instances using RND set-systems properties.

Claim 8.5.7. *With probability $1 - o(1/N)$ in the instance I, simultaneously for all integers $\ell \geq 0$, any collection of ℓ sets from the* fooling instances $\left\{ I^i_j \;\middle|\; i \in [g_r],\ j \in [w_r] \setminus \{j^*\} \right\}$ *can cover at most $\ell \cdot r \cdot (2c \cdot \log N)$ elements in $U^\star$.*

Proof. Recall that for any group $i \in [g_r]$, any instance I^i_j is supported on the set of elements A_j in $\mathcal{A}$ (before applying the labeling function ϕ). Similarly, $U^\star_i$ is the set A_{j^*} (again before applying ϕ). Define $\mathcal{C}_i$ as the collection of all input sets from all players in P_i except the

sets coming from the special instance. By construction, $|\mathcal{C}_i| \leq m_{r-1} \cdot w_r \leq N^{c \cdot r}$ (as $c \geq 4r$). Moreover, for any $j \in [w_r] \setminus \{j^*\}$, since $I^i_j \sim \mathcal{D}'_{r-1}$, by Observation 8.5.3, any member of $\mathcal{C}_i$ is a set of size N chosen uniformly at random from some $A_j \neq A_{j^*}$. This implies that $\mathcal{C}_i$ satisfies the Property (2) in Definition 8.6 (as $\mathcal{A}$ is an (N, r, c)-RND set-system and local mappings of elements are one to one when restricted to the mapping of $\mathcal{X}_i$ to U_i). As such, by definition of an RND set-system, w.p. $1 - o(1/N^3)$, any set $S \in \mathcal{C}$ can cover at most $2c \cdot r \cdot \log N$ elements from $U^\star_i$ and consequently $U^\star$ as $S \cap (U^\star \setminus U^\star_i) = \emptyset$.

We can take a union bound over the $g_r \leq N^2$ different RND set-systems (one belonging to each group) and the above bound holds w.p. $1 - o(1/N)$ for all groups simultaneously. This means that any collection of ℓ sets across any instance I^i_j for $i \in [g_r]$ and $j \neq j^*$, can cover at most $\ell \cdot 2c \cdot r \cdot \log N$ elements in $U^\star$. □ Claim 8.5.7

In the following, we condition on the event in Claim 8.5.7, which happens w.p. at least $1 - 1/N$. Let $\mathcal{C} = \mathcal{C}_s \cup \mathcal{C}_f$ be any collection of k_r sets (i.e., a potential k_r-cover) in the input instance I such that $\mathcal{C}_s$ are $\mathcal{C}_f$ are chosen from the special instances and fooling instances, respectively. Let $\ell_s = |\mathcal{C}_s|$ and $\ell_f = |\mathcal{C}_f|$; we have,

$$
\begin{aligned}
|c(\mathcal{C})| &= |c(\mathcal{C}) \cap U^\star| + |c(\mathcal{C}) \cap (U \setminus U^\star)| \\
&\leq |c(\mathcal{C}_s) \cap U^\star| + |c(\mathcal{C}_f) \cap U^\star| + |U \setminus U^\star| \\
&\leq k_r + \ell_s \cdot (2r - 2) \cdot 2c \cdot \log N + \ell_f \cdot r \cdot 2c \cdot \log N + N^{2r} \\
&\quad \text{(by Claim 8.5.6 for the first term and Claim 8.5.7 for the second term)} \\
&\leq 4k_r + k_r \cdot (2r - 2) \cdot 2c \cdot \log N \qquad (2k_r \geq N^{2r}) \\
&\leq k_r \cdot 2r \cdot 2c \cdot \log N \leq k_r \cdot 2c \cdot \log N^{2r}.
\end{aligned}
$$

This means that w.p. at least $1 - 1/N$, I is also a No instance. □ Lemma 8.5.5

The following claim now follows immediately from Lemmas 8.5.4 and 8.5.5.

Claim 8.5.8. *The packing function σ_r and labeling family Φ_r defined above are γ-preserving for the parameter $\gamma = 1/N$.*

We are now ready to prove Theorem 8.4.

Proof of Theorem 8.4. The results in this section and Claim 8.5.8 imply that the family of distributions $\{\mathcal{D}_r\}_{r=0}^{c}$ for the coverage$(N, r,)$ are γ-hard for the parameter $\gamma = 1/N$, as long as $r \leq 4c \leq 4\sqrt{N/12}$. Consequently, by Theorem 8.3, any r-round protocol that can compute the value of coverage$(N, r,)$ on $\mathcal{D}_r$ w.p. at least $2/3 + r \cdot \gamma = 2/3 + r/N < 3/4$ requires $\Omega(w_r/r^4) = \Omega(N^c/r^4)$ total communication. Recall that the gap between the value of opti-

mal solution between Yes and No instances of coverage(N, r) is at least $N/\left(2c \cdot \log\left(N^{2r}\right)\right) \geq \left(\frac{k_r^{1/2r}}{2c \cdot \log k_r}\right)$. As such, any r-round distributed algorithm that can approximate the value of optimal solution to within a factor better than this w.p. at least $3/4$ can distinguish between Yes and No cases of this distribution, and hence requires $\Omega(N^{c-2r}/r^4) = \Omega\left(\frac{k_r}{r^4} \cdot m^{\frac{c}{(c+2)\cdot 4r}}\right)$ per player communication. Finally, since $N \leq 2k_r^{1/2r}$, the condition $c \leq \sqrt{N/12}$ holds as long as $c = o\left(\frac{\log k_r}{\log\log k_r}\right)$, finalizing the proof. □

8.6. Distributed Algorithms for Maximum Coverage

In this section, we show that both the *round-approximation* tradeoff and the *round-communication* tradeoff achieved by our lower bound in Theorem 8.4 are essentially tight, formalizing Result 8.2.

8.6.1. An $O(r \cdot k^{1/r})$-Approximation Algorithm

Recall that Theorem 8.4 shows that getting better than $k^{\Omega(1/r)}$ approximation in r rounds requires a relatively large communication of $m^{\Omega(1/r)}$, (potentially) larger than any poly(n). In this section, we prove that this round-approximation tradeoff is essentially tight by showing that one can always obtain a $k^{O(1/r)}$ approximation (with a slightly larger constant in the exponent) in r rounds using a limited communication of nearly linear in n.

Theorem 8.5. *There exists a deterministic distributed algorithm for the maximum coverage problem that for any integer $r \geq 1$ computes an $O(r \cdot k^{1/r+1})$ approximation in r rounds and $\widetilde{O}(n)$ communication per each machine.*

On a high level, our algorithm follows an iterative sketching method: in each round, each machine computes a small collection $\mathcal{C}_i$ of its input sets $\mathcal{S}_i$ as a sketch and sends it to the coordinator. The coordinator is maintaining a collection of sets $\mathcal{X}$ and updates it by iterating over the received sketches and picking any set that still has a relatively large contribution to this partial solution. The coordinator then communicates the set of elements covered by $\mathcal{X}$ to the machines and the machines update their inputs accordingly and repeat this process. At the end, the coordinator returns (a constant approximation to) the optimal k-cover over the collection of *all* received sets across different rounds.

In the following, we assume that our algorithm is given a value $\widetilde{\text{opt}}$ such that $\text{opt} \leq \widetilde{\text{opt}} \leq 2 \cdot \text{opt}$. We can remove this assumption by guessing the value of $\widetilde{\text{opt}}$ in powers of two (up to n) and solve the problem simultaneously for all of them and return the best solution, which increases the communication cost by only an $O(\log n)$ factor.

We first introduce the algorithm for computing the sketch on each machine; the algorithm is a simple thresholding version of the greedy algorithm for maximum coverage.

GreedySketch$(U, \mathcal{S}, \tau)$. An algorithm for computing the sketch of each machine's input.

Input: A collection $\mathcal{S}$ of sets from $[n]$, a target universe $U \subseteq [n]$, and a threshold τ.

Output: A collection $\mathcal{C}$ of subsets of U.

1. Let $\mathcal{C} = \emptyset$ initially.
2. Iterate over the sets in $\mathcal{S}$ in an arbitrary order and for each set $S \in \mathcal{S}$, if $|(S \cap U) \setminus c(\mathcal{C})| \geq \tau$, then add $(S \cap U) \setminus c(\mathcal{C})$ to $\mathcal{C}$.
3. Return $\mathcal{C}$ as the answer.

Notice that in the Line (2) of GreedySketch, we are adding the *new contribution* of the set S and not the complete set itself. This way, we can bound the total representation size of the output collection $\mathcal{C}$ by $\widetilde{O}(n)$ (as each element in U appears in at most one set). We now present our algorithm in Theorem 8.5.

Algorithm 2: Iterative Sketching Greedy (ISGreedy).

Input: A collection $\mathcal{S}_i$ of subsets of $[n]$ for each machine $i \in [p]$ and a value $\widetilde{\text{opt}} \in [\text{opt}, 2 \cdot \text{opt}]$.

Output: A k-cover from the sets in $\mathcal{S} := \bigcup_{i \in [p]} \mathcal{S}_i$.

1. Let $\mathcal{X}^0 = \emptyset$ and $U_i^0 = [n]$, for each $i \in [p]$ initially. Define $\tau := \widetilde{\text{opt}}/4r \cdot k$.
2. For $j = 1$ to r rounds:
   (a) Each machine i computes $\mathcal{C}_i^j = \text{GreedySketch}(U_i^{j-1}, \mathcal{S}_i, \tau)$ and sends it to coordinator.
   (b) The coordinator sets $\mathcal{X}^j = \mathcal{X}^{j-1}$ initially and iterates over the sets in $\bigcup_{i \in [p]} \mathcal{C}_i^j$, in decreasing order of $\left|c(\mathcal{C}_i^j)\right|$ over i (and consistent with the order in GreedySketch for each particular i), and adds each set S to $\mathcal{X}^j$ if $\left|S \setminus c(\mathcal{X}^j)\right| \geq \frac{1}{k^{1/r+1}} \cdot |S|$.
   (c) The coordinator communicates $c(\mathcal{X}^j)$ to each machine i and the machine updates its input by setting $U_i^j = c(\mathcal{C}_i^j) \setminus c(\mathcal{X}^j)$.
3. At the end, the coordinator returns the best k-cover among all sets in $\mathcal{C} := \bigcup_{i \in [p], j \in [r]} \mathcal{C}_i^j$ sent by the machines over all rounds.

The round complexity of ISGreedy is trivially r. For its communication cost, notice that at each round, each machine is communicating at most $\widetilde{O}(n)$ bits and the coordinator communicates $\widetilde{O}(n)$ bits back to each machine. As the number of rounds never needs to be more than $O(\log k)$, we obtain that ISGreedy requires $\widetilde{O}(n)$ communication per each machine. Therefore, it only remains to analyze the approximation guarantee of this algorithm. To do so, it suffices to show that,

Lemma 8.6.1. *Define* $\mathcal{C} := \bigcup_{i\in[p],j\in[r]} \mathcal{C}_i^j$. *The optimal* k*-cover of* $\mathcal{C}$ *covers* $\left(\frac{\text{opt}}{4r\cdot k^{1/r+1}}\right)$ *elements.*

Proof. We prove Lemma 8.6.1 by analyzing multiple cases.

Claim 8.6.2. *If* $|\mathcal{X}^r| \geq k$, *then the optimal* k*-cover of* $\mathcal{X}^r \subseteq \mathcal{C}$ *covers* $\left(\frac{\text{opt}}{4r\cdot k^{1/r+1}}\right)$ *elements.*

Proof. Consider the first k sets added to the collection $\mathcal{X}^r$. Any set S that is added to $\mathcal{X}^r$ in (Line (2b) of ISGreedy) covers $\frac{1}{k^{1/r+1}} \cdot |S|$ new elements. Moreover, $|S| \geq \tau = \widetilde{\text{opt}}/4rk$ (by Line (2) of the GreedySketch algorithm). Hence, the first k sets added to $\mathcal{X}^r$ already cover at least,

$$k \cdot \frac{1}{k^{1/r+1}} \cdot \frac{\widetilde{\text{opt}}}{4rk} \geq \frac{\text{opt}}{4r \cdot k^{1/r+1}}$$

elements, proving the claim. □ Claim 8.6.2

The more involved case is when $|\mathcal{X}^r| < k$, which we analyze below. Recall that $\mathcal{C}_i^j$ is the collection computed by GreedySketch$(U_i^{j-1}, \mathcal{S}_i, \tau)$ on the machine $i \in [p]$ in round j. We can assume that each $\left|\mathcal{C}_i^j\right| < k$; otherwise consider the smallest value of j for which the for the first time there exists an $i \in [p]$ with $\left|\mathcal{C}_i^j\right| \geq k$ (if for this value of j, there are more than one choice for i choose the one with the largest size of $c(\mathcal{C}_i^j)$): in Line (2b), the coordinator would add all the sets in $\mathcal{C}_i^j$ to $\mathcal{X}^j$ making $\left|\mathcal{X}^j\right| \geq k$, a contradiction with the assumption that $|\mathcal{X}^r| < k$.

By the argument above, if there exists a machine $i \in [p]$, with $\left|c(\mathcal{C}_i^1)\right| > \text{opt}/4k^{1/r+1}$, we are already done. This is because the collection $\mathcal{C}_i^1$ contains at most k sets and hence $\mathcal{C}_i^1$ is a valid k-cover in $\mathcal{C}$ that covers $(\text{opt}/4k^{1/r+1})$ elements, proving the lemma in this case. It remains to analyze the more involved case when none of the above happens.

Lemma 8.6.3. *Suppose* $|\mathcal{X}^r| < k$ *and* $\left|c(\mathcal{C}_i^1)\right| \leq \text{opt}/4k^{1/r+1}$ *for all* $i \in [p]$*; then, the optimal* k*-cover of* $\mathcal{C}$ *covers* $\left(\frac{\text{opt}}{4r\cdot k^{1/r+1}}\right)$ *elements.*

Proof. Recall that in each round $j \in [r]$, each machine $i \in [p]$ first computes a collection $\mathcal{C}_i^j$ from the universe U_i^{j-1} as its sketch (using GreedySketch) and sends it to the coordinator; at the end of the round also this machine i updates its target universe for the next round to $U_i^j \subseteq \mathcal{C}_i^j$. We first show that this target universe U_i^j shrinks in each round by a large factor compared to $\mathcal{C}_i^j$.

Claim 8.6.4. *For any round* $j \in [r]$ *and any machine* $i \in [p]$, $\left|U_i^j\right| \leq \left(1/k^{1/r+1}\right) \cdot \left|c(\mathcal{C}_i^j)\right|$.

Proof. Consider any $i \in [p]$ and round $j \in [r]$; by Line (2c) of ISGreedy, we know $U_i^j = c(\mathcal{C}_i^j) \setminus c(\mathcal{X}^j)$. Hence, it suffices to show that $\mathcal{X}^j$ covers $(1 - 1/k^{1/r+1})$ fraction of $c(\mathcal{C}_i^j)$. This

is true because for any set $S \in \mathcal{C}_i^j$ that is *not* added to $\mathcal{X}^j$, we have, $\left|S \setminus c(\mathcal{X}^j)\right| < \frac{1}{k^{1/r+1}} \cdot |S|$, meaning that at most $1/k^{1/r+1}$ fraction of any set $S \in \mathcal{C}_i^j$ can remain uncovered by $\mathcal{X}^j$ at the end of the round j. □ Claim 8.6.4

By Claim 8.6.4, and the assumption on size of $\left|c(\mathcal{C}_i^1)\right|$ in the lemma statement, we have,

$$\begin{aligned}
|c(\mathcal{C}_i^r)| \leq \left|U_i^{r-1}\right| &\leq \left(\frac{1}{k^{1/r+1}}\right) \cdot \left|c(\mathcal{C}_i^{r-1})\right| \leq \left(\frac{1}{k^{1/r+1}}\right) \cdot \left|U_i^{r-2}\right| \\
&\text{(since } U_i^j \subseteq c(\mathcal{C}_i^j) \subseteq U_i^{j-1} \text{ by construction of ISGreedy and GreedySketch)} \\
&\leq \left(\frac{1}{k^{1/r+1}}\right)^{r-1} \cdot \left|c(\mathcal{C}_i^1)\right| \leq \left(\frac{1}{k^{1/r+1}}\right)^{r-1} \cdot \frac{\text{opt}}{4k^{1/r+1}} \\
&\text{(by expanding the bound on each } \left|U_i^j\right| \text{ recursively and using the bound on } \left|c(C_i^1)\right|) \\
&\leq \frac{\text{opt}}{4k^{r/r+1}}. \qquad (8.3)
\end{aligned}$$

Fix any optimal solution OPT. We make the sets in OPT *disjoint* by arbitrarily assigning each element in $c(\text{OPT})$ to exactly one of the sets that contains it. Hence, a set $O \in \text{OPT}$ is a subset of one of the original sets in $\mathcal{S}$; we slightly abuse the notation and say O belongs to $\mathcal{S}$ (or input of some machine) to mean that the corresponding super set belongs to $\mathcal{S}$. In the following, we use Eq (8.3) to argue that any set $O \in \text{OPT}$ has a "good representative" in the collection $\mathcal{C}$. This is the key part of the proof of Lemma 8.6.3 and the next two claims are dedicated to its proof.

We first show that for any set O in the optimal solution that belonged to machine $i \in [p]$, if O was never picked in any $\mathcal{X}^j$ during the algorithm, then the universe U_i^j at any step covers a large portion of O. For any $j \in [r]$ and $i \in [p]$, define $X^j := c(\mathcal{X}^j)$ and $C_i^j = c(\mathcal{C}_i^j)$. We have,

Claim 8.6.5. *For any set $O \in \text{OPT} \setminus \mathcal{C}$ and the parameter τ defined in* ISGreedy*, if O appears in the input of machine $i \in [p]$, then, for any $j \in [r]$, $\left|O \cap U_i^j\right| \geq \left|O \setminus X^j\right| - j \cdot \tau$.*

Proof. The idea behind the proof is as follows. In each round j, among the elements already in U_i^{j-1}, at most τ elements of O can be left uncovered by the set C_i^j as otherwise the GreedySketch algorithm should have picked O (a contradiction with $O \notin \mathcal{C}$). Moreover, any element in C_i^j but not U_i^j is covered by $c(\mathcal{X}^j)$ i.e., X^j and hence can be accounted for in the term $\left|O \setminus X^j\right|$.

We now formalize the proof. The proof is by induction. The base case for $j = 0$ is trivially true as $U_i^0 = [n]$ and $X^0 = \emptyset$ (as $\mathcal{X}^0 = \emptyset$). Now assume inductively that this is the case for integers smaller than j and we prove it for j. By Line (2) of GreedySketch, we know $\left|O \cap U_i^{j-1} \setminus C_i^j\right| < \tau$ as otherwise the set O would have been picked

by $\mathsf{GreedySketch}(U_i^{j-1}, \mathcal{S}_i, \tau)$ in $\mathsf{ISGreedy}$, a contradiction with the fact that $O \notin \mathcal{C}$. Using this plus the fact that $C_i^j = c(\mathcal{C}_i^j) \subseteq U_i^{j-1}$, we have,

$$\left|O \cap C_i^j\right| = \left|O \cap C_i^j \cap U_i^{j-1}\right| \geq \left|O \cap U_i^{j-1}\right| - \left|O \cap U_i^{j-1} \setminus C_i^j\right| \geq \left|O \setminus X^{j-1}\right| - j \cdot \tau, \quad (8.4)$$

where the last inequality is by induction hypothesis on the first term and the bound of τ on the second term.

To continue, define $Y^j = X^j \setminus X^{j-1}$, i.e., the set of new elements covered by $\mathcal{X}^j$ compared to $\mathcal{X}^{j-1}$. By construction of the algorithm $\mathsf{ISGreedy}$, $U_i^j = C_i^j \setminus X^j = C_i^j \setminus Y^j$ as U_i^{j-1} and consequently C_i^j do not have any intersection with X^{j-1}. We now have,

$$\begin{aligned}
\left|O \cap U_i^j\right| &= \left|O \cap \left(C_i^j \setminus Y^j\right)\right| \geq \left|O \cap C_i^j\right| - \left|O \cap Y^j\right| \\
&\underset{\text{Eq (8.4)}}{\geq} \left|O \setminus X^{j-1}\right| - j \cdot \tau - \left|O \cap Y^j\right| \\
&= \left|O \setminus \left(X^j \setminus Y^j\right)\right| - j \cdot \tau - \left|O \cap Y^j\right| && \text{(by definition of } Y^j = X^j \setminus X^{j-1}) \\
&= \left|O \setminus X^j\right| - j \cdot \tau, && \text{(since } Y_j \subseteq X_j)
\end{aligned}$$

which proves the induction step. □ Claim 8.6.5

We next argue that since any set $O \in \text{OPT}$ that is located on machine i is "well represented" in U_i^r by Claim 8.6.5 (if not already picked in $\mathcal{X}^r$), and since by Eq (8.3), size of $\mathcal{C}_i^r$ and consequently the number of sets sent by machine i in $\mathcal{C}_i^r$ is small, there should exists a set in $\mathcal{C}_i^r$ that also represents O rather closely. Formally,

Claim 8.6.6. *For any set $O \in \text{OPT}$, there exists a set $S_O \in \mathcal{C}$ such that for the parameter τ defined in $\mathsf{ISGreedy}$, $|O \cap S_O| \geq \frac{|O \setminus X^r| - r \cdot \tau}{r \cdot k^{1/r+1}}$.*

Proof. Fix a set $O \in \text{OPT}$ and assume it appears in the input of machine $i \in [p]$. The claim is trivially true if $O \in \mathcal{C}$ (as we can take $S_O = O$). Hence, assume $O \in \text{OPT} \setminus \mathcal{C}$. By Claim 8.6.5 and the fact that $U_i^r \subseteq C_i^r$, at the end of the last round r, we have,

$$|O \cap C_i^r| \geq |O \cap U_i^r| \underset{\text{Claim 8.6.5}}{\geq} |O \setminus X^r| - r \cdot \tau.$$

Moreover, by Eq (8.3), $|C_i^r| \leq \text{opt}/4k^{r/r+1}$. Since any set added to $\mathcal{C}_i^r$ increases $C_i^r = c(\mathcal{C}_i^r)$ by at least $\tau = \text{opt}/4kr$ elements (by construction of $\mathsf{GreedySketch}$), we know that,

$$|\mathcal{C}_i^r| \leq \frac{|C_i^r|}{\text{opt}/4kr} \underset{\text{Eq (8.3)}}{\leq} r \cdot k^{1/r+1}.$$

It is easy to see that there exists a set $S_O \in \mathcal{C}_i^r$ that covers at least $1/|\mathcal{C}_i^r|$ fraction of $O \cap C_i^r$;

combining this with the equations above, we obtain that,

$$|O \cap S_O| \geq \frac{|O \setminus X^r| - r \cdot \tau}{r \cdot k^{1/r+1}}. \quad \square$$

$\square$ Claim 8.6.6

We are now ready to finalize the proof of Lemma 8.6.3. Define $\mathcal{C}_O := \{S_O \in \mathcal{C} \mid O \in \text{OPT}\}$ for the sets S_O defined in Claim 8.6.6. Clearly, $\mathcal{C}_O \subseteq \mathcal{C}$ and $|\mathcal{C}_O| \leq k$. Additionally, recall that $|\mathcal{X}| < k$ by the assumption in the lemma statement. Consequently, both $\mathcal{C}_O$ and $\mathcal{X}$ are k-covers in $\mathcal{C}$. In the following, we show that the best of these two collections covers $(\text{opt}/4r \cdot k^{1/r+1})$ elements.

$$\begin{aligned}
|c(\mathcal{C}_O)| + |c(\mathcal{X}^r)| &= \left|\bigcup_{O \in \text{OPT}} S_O\right| + |X^r| \geq \left|\bigcup_{O \in \text{OPT}} (O \cap S_O)\right| + |X^r| \\
&= \sum_{O \in \text{OPT}} |O \cap S_O| + |X^r| \\
&\text{(as by the discussion before Claim 8.6.5 we assume the sets in OPT are disjoint)} \\
&\underset{\text{Claim 8.6.6}}{\geq} \sum_{O \in \text{OPT}} \left(\frac{|O \setminus X^r| - r \cdot \tau}{r \cdot k^{1/r+1}}\right) + |X^r| \\
&= \frac{\left|\bigcup_{O \in \text{OPT}} O \setminus X^r\right| - k \cdot r \cdot \tau}{r \cdot k^{1/r+1}} + |X^r| \\
&\text{(again by the assumption on the disjointness of the sets in OPT and the fact that } |\text{OPT}| = k) \\
&\geq \frac{|c(\text{OPT})| - |X^r| - \widetilde{\text{opt}}/4}{r \cdot k^{1/r+1}} + |X^r| \qquad (\text{as } \tau = \widetilde{\text{opt}}/4kr) \\
&\geq \frac{|c(\text{OPT})| - \text{opt}/2}{r \cdot k^{1/r+1}} \geq \frac{\text{opt}}{2r \cdot k^{1/r+1}}. \quad (\text{as } |c(\text{OPT})| = \text{opt and } \widetilde{\text{opt}} \leq 2 \cdot \text{opt})
\end{aligned}$$

As a result, at least one of $\mathcal{C}_O$ or $\mathcal{X}^r$ is a k-cover that covers $(\text{opt}/4r \cdot k^{1/r+1})$ elements, finalizing the proof. $\square$ Lemma 8.6.3

Lemma 8.6.1 now follows immediately from Claim 8.6.2 and Lemma 8.6.3. $\square$ Lemma 8.6.1

Theorem 8.5 follows from Lemma 8.6.1 as the coordinator can simply run any constant factor approximation algorithm for maximum coverage on the collection $\mathcal{C}$ and obtains the final result.

8.6.2. An $(\frac{e}{e-1})$-Approximation Algorithm

We now prove that the *round-communication* tradeoff for the distributed maximum coverage problem proven in Theorem 8.4 is essentially tight. Theorem 8.4 shows that using $k \cdot m^{O(1/r)}$ communication in r rounds only allows for a relatively large approximation factor of $k^{\Omega(1/r)}$.

Here, we show that we can always obtain an (almost) $\left(\frac{e}{e-1}\right)$-approximation (the optimal approximation ratio with sublinear in m communication) in r rounds using $k \cdot m^{\Omega(1/r)}$ (for some larger constant in the exponent).

As stated in the introduction, our algorithm in this part is quite general and works for maximizing any monotone submodular function subject to a cardinality constraint (see Section 2.5.1 for definitions). Hence, in the following, we present our results in this more general form.

Theorem 8.6. *There exists a randomized distributed algorithm for submodular maximization subject to cardinality constraint that for any ground set V of size m, any monotone submodular function $f : 2^V \to \mathbb{R}^+$, and any integer $r \geq 1$ and parameter $\varepsilon \in (0,1)$, with high probability computes an $\left(\frac{e}{e-1}+\varepsilon\right)$-approximation in r rounds while communicating $O(k \cdot m^{O(1/\varepsilon \cdot r)})$ items from V.*

We assume that the algorithm is given a value $\widetilde{\text{opt}}$ such that $\text{opt} \leq \widetilde{\text{opt}} \leq 2 \cdot \text{opt}$. In general, one can guess $\widetilde{\text{opt}}$ in powers of two in the range Δ to $k \cdot \Delta$ in parallel and solve the problem for all of them and return the best solution. This would increase the communication cost by only a factor of $\Theta(\log k)$ (and one extra round of communication just to communicate Δ if it is unknown). We now present our algorithm.

Algorithm 1: Sample and Prune Greedy (SPGreedy).

Input: A collection $V_i \subseteq V$ of items for each machine $i \in [p]$ and a value $\widetilde{\text{opt}} \in [\text{opt}, 2 \cdot \text{opt}]$.

Output: A collection of k items from V.

1. Define the parameters $\ell := \left\lceil \lg_{(1+\varepsilon)}(2e) \right\rceil (= \Theta(1/\varepsilon))$ and $s = \lceil r/\ell \rceil$. The algorithm consists of ℓ *iterations* each with s *steps*.
2. For $j = 1$ to ℓ iterations:
   (a) Let $\tau_j = \frac{\widetilde{\text{opt}}}{k} \cdot \left(\frac{1}{1+\varepsilon}\right)^{j-1}$ and $X^{j,0} = X^{j-1,s}$ initially (we assume $X^{0,*} = \emptyset$).
   (b) For $t = 1$ to s steps:
      i. Define $V^{j,t} = \{a \in V \mid f_{X^{j,(t-1)}}(a) \geq \tau_j\}$.
      ii. Each machine $i \in [p]$ samples each item in $V^{j,t} \cap V_i$ independently and with probability $q_t := \begin{cases} \frac{4k \log m}{m^{1-(t/s)}} & \text{if } t < s \\ 1 & \text{if } t = s \end{cases}$, and sends them to the coordinator.
      iii. The coordinator iterates over each received item a (in an arbitrary order) and adds a to $X^{j,t}$ iff $f_{X^{j,t}}(a) \geq \tau_j$.
      iv. The coordinator communicates the set $X^{j,t}$ to the machines.

3. The coordinator returns $X^{\ell,s}$ in the last step (if at any earlier point of the algorithm size of some $X^{*,*}$ is already k, the coordinator terminates the algorithm and outputs this set as the answer).

SPGreedy requires $\ell = \Theta(1/\varepsilon)$ iterations each consists of $s = \lceil r/\ell \rceil$ steps. Moreover, each step can be implemented in one round of communication. As such, the round complexity of this algorithm is simply $O(r)$. In the following, we prove a bound on the communication cost of this algorithm and then analyze its approximation guarantee. To do so, we need the following auxiliary lemma on the size of each set $V^{j,t}$ in the algorithm.

Lemma 8.6.7. *For any $j \in [\ell]$ and any $t \in [s]$, $\left|V^{j,t}\right| \leq m^{1-(t-1)/s}$ w.p. at least $1 - 1/m^{2k}$.*

Proof. Fix any iteration $j \in [\ell]$ and observe that $X^{j,0} \subseteq \ldots \subseteq X^{j,s}$. By submodularity of $f(\cdot)$, this means that for $a \in V$, $f_{X^{j,0}}(a) \geq \ldots \geq f_{X^{j,s}}(a)$ and so $V^{j,1} \supseteq V^{j,2} \supseteq \ldots \supseteq V^{j,s}$.

The bound in the lemma statement is trivially true for $t = 1$; hence, we prove it for any $t > 1$. To do so, we show that the collection $X^{j,t-1}$, computed at the end of the $(t-1)$-th step in iteration j, has the property that the corresponding collection $V^{j,t}$ (which is uniquely identified by $X^{j,t}$) has its size bounded as in the lemma statement.

Fix any set A of up to k items from V. We say that A is *bad* iff the set $V_A := \{a \in V \mid f_A(a) \geq \tau_j\}$ has size more than $m^{1-(t-1)/s}$. For the set $X^{j,t}$ to be equal to A at the end of the $(t-1)$-th step (in iteration j), necessarily no item from V_A should be sampled by any of the machines in that round. As such, for any bad set $A \subseteq V$,

$$\Pr\left(X^{j,t} = A\right) \leq (1 - q_{t-1})^{|V_A|} \leq \left(1 - \frac{4k \cdot \log m}{m^{1-(t-1)/s}}\right)^{m^{1-(t-1)/s}} \leq \exp\left(-4k \cdot \log m\right) \leq m^{-4k}.$$

Taking a union bound over $\sum_{i=1}^{k} \binom{m}{i} = O(m^k)$ possible choices for a bad set A, the probability that any bad set A is chosen as the set $X^{j,t}$ is smaller than $1/m^{3k}$. Conditioned on this event, the set $V_{X^{j,t}}$ for the next round, i.e., the t-th round, has size at most $m^{1-(t-1)/s}$. Taking a union bound over all $j \in [\ell]$ and $t \in [s]$ finalizes the proof. □

It is now easy to bound the communication cost of this protocol.

Lemma 8.6.8. SPGreedy *communicates $O(r \cdot k \cdot m^{1/s} \cdot \log m)$ items w.p. at least $1 - 1/m^k$.*

Proof. We condition on the event in Lemma 8.6.7. As such, for each iteration $j \in [\ell]$ and each step $t \in [s]$ in this iteration, $V^{j,t}$ is of size $m^{1-(t-1)/s}$ at most. Consequently, the total number of items sampled by the machines in step t is in expectation at most

$m^{1-(t-1)/s} \cdot q_t = m^{1/s} \cdot 4k \log m$. This means that, by Chernoff bound, w.p. at least $1 - 1/m^{2k}$, at most $O(m^{1/s} \cdot k \log m)$ items are communicated by each machine in this step. The coordinator also communicates at most k items to each machine in each step. The bound now follows by taking a union bound over all $O(r)$ iterations and steps. □

Lemma 8.6.9. *Suppose X is the set returned by* SPGreedy*; then, $f(X) \geq (1-1/e-\varepsilon) \cdot \text{opt}$.*

Proof. We first argue that if the set X has size $< k$ then $f(X) \geq (1-1/e) \cdot \text{opt}$ already; note that in this case, $X = X^{\ell,s}$. Let OPT be an optimal solution and consider any item $o \in \text{OPT}$ that was never picked by the coordinator to be added to X; this in particular means that o was not added to $X^{\ell,s}$ which implies,

$$f_X(o) = f_{X^{\ell,s}}(o) < \tau_\ell = \frac{\widetilde{\text{opt}}}{k} \cdot \left(\frac{1}{1+\varepsilon}\right)^{\ell} = \frac{\widetilde{\text{opt}}}{2e \cdot k} \leq \frac{\text{opt}}{e \cdot k}. \tag{8.5}$$

The first inequality in Eq (8.5) holds because in step s of each iteration, *every* item $a \in V$ with $f_{X^{\ell,s-1}}(a) \geq f_{X^{\ell,s}}(a)$ (by submodularity) is sent to the coordinator and hence if $f_{X^{\ell,s}}(o) \geq \tau_\ell$ the coordinator would be able to find it and add it to $X^{\ell,s}$. The next two equalities are by the choices of τ_ℓ and ℓ, respectively, and the last inequality is true since $\widetilde{\text{opt}} \leq 2\text{opt}$. Using this bound and the monotone submodularity of $f(\cdot)$, we can write,

$$f(\text{OPT}) \underset{\text{Fact 2.5.2}}{\leq} f(X) + \sum_{o \in \text{OPT} \setminus X} f_X(o) \underset{\text{Eq (8.5)}}{\leq} f(X) + |\text{OPT}| \cdot \frac{\text{opt}}{e \cdot k} = f(X) + \text{opt}/e$$

as $|\text{OPT}| = k$, which finalizes the proof in this case.

We now consider the more involved case where the coordinator picks exactly k items in X. To continue, we need the following definitions. Let $x_1, \ldots, x_k$ be the items added to X by the coordinator in this particular order. For any $i \in [k]$, define $X^{<i} = x_1, \ldots, x_{i-1}$, i.e., the first $i-1$ items added to X (define $X^{<1} = \emptyset$). We have,

Claim 8.6.10. *For any $i \in [k]$, $f_{X^{<i}}(x_i) \geq \frac{f(\text{OPT}) - f(X^{<i})}{(1+\varepsilon) \cdot k}$.*

Proof. For any item x_i for $i \in [k]$, by construction of SPGreedy, if i is added in iteration $j \in [\ell]$ to X, then,

$$f_{X^{<i}}(x_i) \geq \tau_j. \tag{8.6}$$

Suppose first that the item x_i is added to X in the first iteration. By the above equation,

$$f_{X^{<i}}(x_i) \underset{\text{Eq (8.6)}}{\geq} \tau_1 = \frac{\widetilde{\text{opt}}}{(1+\varepsilon) \cdot k} \geq \frac{\text{opt}}{(1+\varepsilon) \cdot k},$$

by the bounds on τ_1 and $\widetilde{\text{opt}}$. This proves the lemma for any item x_i that is added to X in the first iteration. Now suppose x_i is added in the iteration $j > 1$.

Consider the item $o^* \in \text{OPT} \setminus X^{<i}$ with the maximum marginal contribution to $f_{X^{<i}}$. Recall that since $f(\cdot)$ is submodular, by Fact 2.5.3, $f_{X^{<i}}(\cdot)$ is subadditive. We have,

$$f_{X^{<i}}(o^*) = \max_{o \in \text{OPT} \setminus X^{<i}} f_{X^{<i}}(o) \underset{\text{Fact 2.5.3}}{\geq} \frac{1}{k} \cdot \sum_{o \in \text{OPT} \setminus X^{<i}} f_{X^{<i}}(o) \underset{\text{Fact 2.5.2}}{\geq} \frac{1}{k} \cdot \left(f(\text{OPT}) - f(X^{<i})\right) \tag{8.7}$$

On the other hand, we also know that o^* does not belong to $X^{<i}$, meaning that it was not added to $X^{<i}$ at least by end of iteration $j-1$ (since x_i is added to $X^{<i}$ in iteration j). Hence, again by construction of SPGreedy, similar to the case in Eq (8.6),

$$f_{X^{<i}}(o^*) < \tau_{j-1}. \tag{8.8}$$

Finally,

$$f_{X^{<i}}(x_i) \underset{\text{Eq (8.6)}}{\geq} \tau_j \geq \frac{1}{(1+\varepsilon)} \cdot \tau_{j-1} \underset{\text{Eq (8.8)}}{\geq} \frac{1}{(1+\varepsilon)} \cdot f_{X^{<i}}(o^*) \underset{\text{Eq (8.7)}}{\geq} \frac{f(\text{OPT}) - f(X^{<i})}{(1+\varepsilon) \cdot k},$$

finishing the proof. □ Claim 8.6.10

We can now finalize the proof of Lemma 8.6.9 as follows,

$$\begin{aligned}
f(\text{OPT}) - f(X) &= f(\text{OPT}) - f(X^{<k}) - f_{X^{<k}}(x_k) && \text{(by definition of } f_{X^{<k}}(x_k)\text{)} \\
&\underset{\text{Claim 8.6.10}}{\leq} f(\text{OPT}) - f(X^{<k}) - \frac{f(\text{OPT}) - f(X^{<k})}{(1+\varepsilon) \cdot k} \\
&= \left(1 - \frac{1}{(1+\varepsilon) \cdot k}\right) \cdot \left(f(\text{OPT}) - f(X^{<k})\right) \\
&\leq \left(1 - \frac{1}{(1+\varepsilon) \cdot k}\right)^k \cdot \left(f(\text{OPT}) - f(X^{<1})\right) \\
& && \text{(by applying Claim 8.6.10 recursively)} \\
&\leq (1/e + \varepsilon) \cdot f(\text{OPT})
\end{aligned}$$

as $f(X^{<1}) = 0$ since $X^{<1} = \emptyset$ by definition. Thus, $f(X) \geq (1 - 1/e - \varepsilon) \cdot \text{opt}$. □ Lemma 8.6.9

Theorem 8.6 now follows immediately from Lemma 8.6.8 and Lemma 8.6.9.

We conclude this section by stating the following corollary of Theorem 8.6 for the maximum coverage problem, which formalizes the first part of Result 8.2. The proof is a

direct application of Theorem 8.6 plus the known sketching methods for coverage functions in [247, 50] to further optimize the communication cost. Hence we omit the proof here and refer the interested reader to our paper [31] for the proof.

Corollary 8.7. *There exists a distributed algorithm for the maximum coverage problem that for any integer $r \geq 1$, and any parameter $\varepsilon \in (0,1)$, with high probability computes an $\left(\frac{e}{e-1}+\varepsilon\right)$-approximation in r rounds and $\widetilde{O}(\frac{k}{\varepsilon^4} \cdot m^{O(1/\varepsilon \cdot r)} + n)$ total communication.*

8.7. Applications to Other Models of Computation

We discuss the applications of our results to maximum coverage (and submodular maximization) in the dynamic streaming model and the MPC model. We finish the section by making a remark about the role of partitioning of the input in the distributed model.

Maximum coverage in dynamic set streams. By applying the reduction of [14] to dynamic streaming algorithms (Proposition 2.7.11) for maximum coverage problem and using our lower bound in Theorem 8.4 (which was proven in this more general communication model), we obtain that,

Corollary 8.8. *No p-pass semi-streaming algorithm for the maximum coverage problem in the dynamic streaming model can approximate the value of optimal solution to a factor of $o(\frac{k^{1/2p}}{p \cdot \log k})$ with a sufficiently large probability.*

We remark that one can obtain the same exact bounds in Theorem 8.4 for the space complexity of dynamic streaming algorithms also; however, as our focus is on semi-streaming algorithms we provide the above theorem which is qualitatively similar but is easier to parse. Our upper bound in Theorem 8.6, i.e., the SPGreedy algorithm can be implemented in dynamic streams using ℓ_0-samplers (Lemma 3.3.1), implying the following corollary.

Corollary 8.9. *There exists a randomized semi-streaming algorithm for the maximum coverage problem that for any constant $\varepsilon \in (0,1)$, with high probability, computes an $\left(\frac{e}{e-1}+\varepsilon\right)$-approximation in $O(\log m/\varepsilon)$ passes over the stream.*

Corollary 8.9 can also can be stated for dynamic streaming algorithms with different space bounds corresponding to Corollary 8.7; however, for brevity, we only focused on semi-streaming algorithms.

Maximum coverage in the MPC model. We now present our results for maximum coverage and submodular maximization in the MPC model.

Recall that in the sketch-and-update approach (described in Section 8.2) in the MPC model, in each round, every machine is sending a message directly to a designated central

machine for combining the sketches. By definition of the MPC model, the total messages received by the central machine can only be proportional to its memory which is of size $O(s)$. This enforces an upper bound on the total communication of $O(s)$ in each round by the machines. It is thus easy to see that efficient MPC algorithms in the sketch-and-update framework immediately imply communication efficient protocols in the distributed coordinator model (note that this is in general is not true for every MapReduce algorithm). As a result, we can interpret Theorem 8.4 as proving a lower bound for sketch-and-update algorithms in the MapReduce framework.

Corollary 8.10. *For any $\delta \in (0,1)$, any MPC algorithm in the sketch-and-update framework described in Section 8.2 that uses $s = m^{\delta}$ space per machine and computes a constant factor approximation to maximum coverage requires $\Omega(\frac{1}{\delta})$ rounds of communication.*

Moreover, both algorithms in Result 8.2 can be implemented in the MapReduce model. In particular, we state the following corollary of Theorem 8.6 for submodular maximization which also subsumes the results for coverage maximization.

Corollary 8.11. *Let V be a universe of m items and $f : 2^V \to \mathbb{R}^+$ be a monotone submodular function. For any $\varepsilon, \delta \in (0,1)$, there exists an $\left(\frac{e}{e-1} + \varepsilon\right)$-approximation randomized algorithm for maximizing f subject to a cardinality constraint in the MapReduce framework that uses $p = O(m^{1-\delta/\varepsilon})$ machines each with $s = O(m^{\delta/\varepsilon})$ memory and computes the answer in $O(\frac{1}{\varepsilon \cdot \delta})$ rounds.*

Adversarial vs random partitions. We considered *adversarial* input partitions in this work, meaning that the input across the machines is distributed adversarially. However, recall that in Chapter 4, we saw that random partitioning of input can signficantly help in some scenarios. This was in fact first observed for the submodular maximization problem in [249, 112]. For maximum coverage (and submodular maximization), it was shown previously that under this assumption one can achieve a constant factor approximation using $\widetilde{O}(n)$ communication per machine in only *one* round of communication [249, 112]. Comparing this with Theorem 8.4 implies that an approximation factor that can be achieved in only one round of communication and $\widetilde{O}(n)$ communication under randomized partitions, *cannot* be achieved in $o(\frac{\log n}{\log \log n})$ rounds of communication and $\text{poly}(n)$ communication in adversarial partitions! This suggests a remarkable power of randomized composable coresets (and hence our framework in Chapter 4) over adversarial partitions.

Part III

Applications to Resource Constrained Optimization Problems

Chapter 9
Interaction in Combinatorial Auctions

We now start the third and last part of our thesis. In this chapter and next one, we deviate from our main theme of optimization on massive datasets and instead consider settings where the goal is to perform computation over data which is not particularly large but still imposes restrictions of similar nature. We use the toolkit developed in the first two parts of the thesis to obtain several new results for problems of this nature, settling multiple open questions in the literature. We start with studying the role of interaction between players in a combinatorial auction. The materials in this part are based on [25] with a simplification of using our framework in Chapter 8 instead of a direct argument to prove the main result.

The study of role of interaction in combinatorial auctions was originally introduced by Dobzinski, Nisan, and Oren [119] as the following simple market scenario: m items are to be allocated among n bidders in a distributed setting where bidders valuations are private and hence communication is needed to obtain an efficient allocation. The communication happens in rounds: in each round, each bidder, simultaneously with others, broadcasts a message to all parties involved. At the end, the central planner computes an allocation solely based on the communicated messages.

Dobzinski *et al.* [119] made the first progress on this problem by proving that no non-interactive mechanism can solve this problem efficiently, while $O(\log m)$ rounds of interaction suffices to do so. Later, Alon, Nisan, Raz, and Weinstein [19] studied the *qualitatively similar but technically disjoint* setting of bidders with unit-demand valuations and proved that $\Omega(\log \log m)$ rounds of interactions are necessary in this case. Both works posed the following as an open question:

> *What is the "right" level of interaction needed to find an efficient allocation, namely a constant or a poly-log approximation, in combinatorial auctions?*

We resolve this fascinating question by providing an almost tight *round-approximation* tradeoff for this problem, when the players are communicating only polynomially many bits (in n and m). As a corollary, we prove that $\Omega(\frac{\log m}{\log \log m})$ rounds of interaction are *necessary* for obtaining any efficient allocation (i.e., a constant or even a polylog(m)-approximation) in these markets. Our proof of this result builds on the similarity between the setting of this problem and the distributed communication model we study in this thesis and the framework we provided in Chapter 8.

Highlights of Our Contributions

In this chapter, we will establish:

- A tight lower bound on the tradeoff between the number of rounds of interaction combinatorial auctions and the welfare guarantee of the resulting allocation (Section 9.4).

Our lower bound in this chapter uses our framework for proving bounded-round communication complexity lower bounds in the multi-party setting from Chapter 8.

9.1. Background

In a combinatorial auction, m items in M are to be allocated between n bidders (or players[1]) in N with valuation functions $v_i : 2^M \to \mathbb{R}_+$. The goal is to find a collection of disjoint bundles $A_1, \ldots, A_n$ of items in M (an *allocation*), that maximizes *social welfare* defined as the sum of bidder's valuations for the allocated bundles, i.e., $\sum_{i \in N} v_i(A_i)$. We study the tradeoff between the amount of interaction between the bidders and the efficiency of the allocation in combinatorial auctions.

In our model, each bidder $i \in N$ only knows the valuation function v_i and hence the bidders need to communicate to obtain an efficient allocation. Communication happens in rounds. In each round, each bidder i, *simultaneously* with others, broadcasts a message to all parties involved, based on the valuation function v_i and messages in previous rounds. In the last round, the central planner outputs the allocation solely based on the communicated messages. Notice that a "trivial solution" in this setting is for all players to communicate their entire input to the central planner who can then compute an efficient allocation; however, such a protocol is clearly infeasible in most settings as it has an enormous communication cost. As such, we are interested in protocols with significantly less communication cost, typically *exponentially smaller* than the input size.

This model was first introduced by Dobzinski, Nisan, and Oren [119] to address the following fundamental question in economics:

> *"To what extent is interaction between individuals required in order to efficiently allocate resources between themselves?"*

They considered this problem for two different classes of valuation functions: *unit-demand* valuations and *subadditive* valuations. For both settings, they showed that (at least some) interaction is necessary to obtain an efficient allocation: non-interactive (aka 1-round or simultaneous) protocols have enormous communication cost compared to interactive ones, while even allowing a modest amount of interaction allows for finding an (approximately) efficient allocation. We now elaborate more on these results.

For the case of matching markets with n unit-demand bidders and n items (and hence input-size of n bits per each player), Dobzinski *et al.* [119] proved a lower bound of $\Omega(\sqrt{n})$ on the approximation ratio of any simultaneous protocol that communicates $n^{o(1)}$ bits per

[1] Throughout the paper, we use the terms "bidder" and "player" interchangeably.

each bidder. On the other hand, they showed that for any $r \geq 1$, there exists an r-round protocol that achieves an $O(n^{1/r+1})$ approximation by sending $O(\log n)$ bits per each bidder in each round. For the more general setting of combinatorial auctions with n subadditive bidders and m items (and hence input-size of $\exp(m)$ bits per each player), they showed that the best approximation ratio achievable by simultaneous protocols with $\text{poly}(m, n)$ communication is $\Omega(m^{1/4})$, while for any $r \geq 1$, there exists r-round protocols that achieve an approximation ratio of $\widetilde{O}(r \cdot m^{1/r+1})$. These results imply that in such markets, logarithmic rounds of interaction in the market size *suffice* to obtain an (almost) efficient allocation, i.e., a $\text{polylog}(m)$-approximation.

A natural question left open by [119] was to identify the amount of interaction *necessary* to obtain an efficient allocation in these markets. Recently, Alon, Nisan, Raz, and Weinstein [19] provided a partial answer to this question for matching markets: for any $r \geq 1$, any r-round protocol for unit-demand bidders in which each bidder sends at most $n^{o(1)}$ bits in each round can only achieve an $\Omega(n^{1/5^{r+1}})$ approximation [19]. This implies that at least $\Omega(\log \log n)$ rounds of interaction is necessary to achieve an efficient allocation in matching markets. Alon *et al.* [19] further conjectured that the "correct" lower bound for the convergence rate in this setting is $\Omega(\log n)$; in other words, $\Omega(\log n)$ rounds of interaction are necessary for achieving an efficient allocation.

Despite this progress for matching markets, the best known lower bounds for the more general setup of combinatorial auctions with subadditive bidders remained the aforementioned 1-round lower bound of [119], and a $(2-\varepsilon)$-approximation (for every constant $\varepsilon > 0$) for any polynomial communication protocol with unrestricted number of rounds [120]. Indeed, obtaining better lower bounds for r-round protocols was posed as an open problem by Alon *et al.* [19] who also mentioned that: "from a communication complexity perspective, lower bounds in this setup are more compelling, since player valuations require exponentially many bits to encode, hence interaction has the potential to reduce the overall communication from exponential to polynomial.".

9.2. Our Results and Techniques

We resolve the aforementioned open question of Dobzinski *et al.* [119] and Alon *et al.* [19] by proving a tight *round-approximation* tradeoff for subadditive combinatorial auctions.

Result 9.1. *For any $r \geq 1$, any r-round protocol (deterministic or randomized) for combinatorial auctions with* subadditive *bidders that uses polynomial communication in m and n can only achieve an approximation ratio of $\Omega(\frac{1}{r} \cdot m^{1/\Theta(r)})$ to the social welfare.*

We remark that this lower bound holds *even* when the bidders valuations are XOS

functions, a strict subclass of subadditive valuations (see Section 2.5 for definition).

Our main result, combined with the upper bound result of [119], provides a near-complete understanding of the power of each additional round in improving the quality of the allocation in combinatorial auctions. Moreover, a corollary of our result is that in these markets, $\Omega(\frac{\log m}{\log \log m})$ rounds of interaction are *necessary* to achieve any efficient allocation (i.e., constant or poly-log approximation), which is *tight* up to an $O(\log \log m)$ factor.

Our techniques. We use the framework developed in Chapter 8 to prove a communication lower bound in the multi-party communication model with coordinator (which is essentially the same model as the distributed model for combinatorial auctions in this chapter).

Remark 9.2.1. *We shall note that even though in this thesis we use our framework in Chapter 8 to prove our results, chronologically, our results in this part [25] predates our framework in [31] and in fact had a crucial role in designing the framework itself.*

9.2.1. Further Related Work

Communication complexity of combinatorial auctions has received quite a lot of attention in the literature. It is known that for *arbitrary valuations*, exponential amount of communication is needed to obtain an $\left(m^{1/2-\varepsilon}\right)$-approximate allocation (for every constant $\varepsilon > 0$) [259] (see also [258]), and this is also tight [6, 236, 78, 226]. For *subadditive valuations*, a constant factor approximation to the social welfare can be achieved in our model using only polynomial communication [120, 121, 136, 138, 227, 296, 124] (and polynomially many rounds of interaction); in particular, Feige [136] developed a 2-approximation polynomial communication protocol for this problem and Dobzinski, Nisan, and Schapira [120] proved that obtaining $(2-\varepsilon)$-approximation (for any constant $\varepsilon > 0$) requires exponential communication (regardless of the number of rounds).

9.2.2. Subsequent Work

Subsequent to the conference version of this paper [25], Braverman and Oshman [73] improved the lower bounds of Alon *et al.* [19] for *unit-demand markets* to an $\Omega(\frac{\log n}{\log \log n})$ rounds of interaction. While this result is qualitatively similar to our result in this paper for (subadditive) combinatorial auctions, our result and that of [73] are completely *incomparable* in that neither implies or strengthens the other. We also point out that in terms of techniques, our paper and [73] are almost completely disjoint.

9.3. Preliminaries

Intersecting families. We use the following combinatorial construction in our proofs.

Definition 9.1. *A* (p, q, t, ℓ)-intersecting family $\mathcal{F}$ *is a collection of* p *subsets of* $[q]$ *each of*

size t, such that for any two distinct sets $S, T \in \mathcal{F}$, $|S \cap T| \leq \ell$.

An existence of an exponentially large intersecting family with a small pair-wise intersection can be shown by a simple probabilistic argument.

Lemma 9.3.1. *For any integer $r \geq 1$, any parameter $\varepsilon > 0$, and any integer $k \geq \left(2e^2 \cdot r^2\right)^{\frac{1}{\varepsilon}}$, there exists a (p, q, t, ℓ)-intersecting family with $p = \exp\left(\Theta(k^{2r-2+\varepsilon})\right)$, $q = k^{2r} + r \cdot k^{2r-1}$, $t = r \cdot k^{2r-1}$, and $\ell = k^{2r-2+\varepsilon}$.*

Proof. Let $\mathcal{F}$ be a family of p sets (for p to be determined later), each chosen independently and uniformly at random from all t-subsets of $[q]$. Fix any pair of sets $S, T \in \mathcal{F}$; for each element $a \in S$, define the random variable $X_a \in \{0, 1\}$ which is 1 iff $a \in T$ also. We have $\mathbb{E}[X_a] \leq r/k$. Let $X = \sum_{a \in S} X_a$ denotes $|S \cap T|$; hence $\mathbb{E}[X] = r^2 \cdot k^{2r-2}$. Since X_a's are negatively correlated random variables, by Chernoff bound (Proposition 7.3.1),

$$\Pr\left(|S \cap T| > \ell\right) = \Pr\left(X > k^{2r-2+\varepsilon}\right) = \Pr\left(X > k^{\varepsilon}/r^2 \cdot \mathbb{E}[X]\right) \leq \exp\left(-\Omega(k^{2r-2+\varepsilon})\right).$$

By a union bound over all possible choices for $S, T \in \mathcal{F}$,

$$\Pr\left(\exists\ S, T \in \mathcal{F} : |S \cap T| > \ell\right) \leq \sum_{S \neq T \in \mathcal{F}} \Pr\left(|S \cap T| > \ell\right) \leq \binom{p}{2} \cdot \exp\left(-\Omega(k^{2r-2+\varepsilon})\right).$$

Taking $p = \exp\left(\Theta(k^{2r-2+\varepsilon})\right)$ ensures that with some non-zero probability, the set $\mathcal{F}$ is a (p, q, t, ℓ)-intersecting family, implying the existence of such a family. □

Approximation guarantee. We consider protocols that are required to estimate the *maximum value of social welfare* in any instance I of a combinatorial auction (denoted by $\mathsf{sw}(I)$). More formally, a δ-error α-approximation protocol needs to, for each input instance I sampled from $\mathcal{D}$, output a number in the range $[\frac{1}{\alpha} \cdot \mathsf{sw}(I), \mathsf{sw}(I)]$ w.p. at least $1 - \delta$, where the randomness is over the distribution $\mathcal{D}$ (and the randomness of protocol in case of randomized protocols).

Any r-round protocol for finding an approximate allocation can be used to obtain an $(r + 1)$-round protocol for estimating the value of social welfare with $O(n)$ additional communication; simply compute the approximate allocation in the first r rounds and spend one additional round in which each player declares her value for the assigned bundle. It was shown very recently in [71] that this loss of one round in the reduction is unavoidable. However, this extra one round is negligible for our purpose as we are interested in the *asymptotic* dependence of the approximation ratio and number of rounds.

9.4. The Lower Bound

In this section, we establish our main result, formalizing Result 9.1.

Theorem 9.2. *For any integer $1 \leq r \leq o\left(\frac{\log m}{\log\log m}\right)$, and any sufficiently small constant $\varepsilon > 0$, any r-round protocol (possibly randomized) for combinatorial auctions with subadditive (even XOS) bidders that can approximate the value of social welfare to a factor of $\left(\frac{1}{r} \cdot m^{\frac{1-\varepsilon}{2r+1}}\right)$ requires $\exp\left(m^{\Omega(\frac{\varepsilon}{r})}\right)$ bits of communication.*

We start by introducing the recursive family of hard input distributions that we use in proving Theorem 9.2 and then establish a lower bound for this distribution.

A hard input distribution for r-round protocols. Let k be an integer and consider a set N of $n_r = k^{2r}$ players and a set M of $m_r = (r+1) \cdot k^{2r+1}$ items. The players are partitioned (arbitrarily) between k^2 groups $N_1, \ldots, N_{k^2}$ each of size n_{r-1}. Fix a group N_g and for any player $i \in N_g$, we create an exponentially large (in k) collection $\mathcal{C}_i$ of item-sets of size m_{r-1} (over the universe M), such that the for any two sets $S, T \in \mathcal{C}_i$, $|S \cap T| \leq k^{2r-2+\varepsilon}$ (for any constant $\varepsilon > 0$).

The *local* view of player $i \in N_g$ is as follows: over each set $S_j \in \mathcal{C}_i$, we create an $(r-1)$-round instance of the problem, namely instance $I_{i,j}$, sampled from the distribution $\mathcal{D}_{r-1}$ with the set of players N_g and the set of items S_j, and then let the input of player i be the collective input of the i-th player in all these instances. In other words, player i finds herself "playing" in exponentially many "$(r-1)$-round instances" of $\mathcal{D}_{r-1}$.

On the *group level*, the input to players inside a group N_g are highly correlated: for each player $i \in N_g$, one of the instances, namely I_{i,j^*}, is an "special instance" in the sense that *all* players in the group N_g has a "consistent" view of this instance, i.e., the collective view of players $1, \ldots, n_{r-1}$ in N_g on the instances $I_{1,j^*}, \ldots, I_{n_{r-1},j^*}$ forms a valid instance sampled from $\mathcal{D}_{r-1}$. However, for any other index $j \neq j^*$, the collective view of players in N_g in the instances $I_{1,j^*}, \ldots, I_{n_{r-1},j^*}$ forms a "pseudo instance" that is *not* sampled from $\mathcal{D}_{r-1}$; these pseudo instances are created by sampling the input of each player *independently* according to $\mathcal{D}_{r-1}$. Note however that while the pseudo instances and the special instance of a player are fundamentally different, each player is oblivious to this difference, i.e., cannot determine (locally) which instance is the special instance.

Finally, the input to players across the groups, i.e., the *global* input, is further correlated: the set of items in the special instances of players in a group N_g is a "unique" set of items (across all groups), while *all* other instances, across all groups, are constructed over a set of k^{2r} "shared" items. This correlation makes the special instance of a player i, in some sense, the *only important* instance: to obtain a large allocation, the players need to ultimately

solve the problem for these special instances.

We now formally define distribution $\mathcal{D}_r$. In the following, for simplicity of exposition, we assume that the distribution $\mathcal{D}_r$, in addition to the valuation function of players, also outputs the *private collections* (defined similarly as in $\mathcal{D}_1$) of players that are used to define these functions.

Distribution $\mathcal{D}_r(N, M)$. A hard input distribution for r-round protocols (for $r \geq 1$).

Input: Collections N of $n_r = k^{2r}$ players and M of $m_r = (r+1) \cdot k^{2r+1}$ items.

Output: A set of n_r valuation functions $(v_1, \ldots, v_{n_r})$ for the players in N and n_r private collections $(\mathcal{F}_1, \ldots, \mathcal{F}_{n_r})$ used to define the valuation functions.

---

1. Let $\mathcal{S}_r = \{S_1, \ldots, S_p\}$ be a (p_r, q_r, t_r, ℓ_r)-intersecting family with parameters $p_r = p = \exp(\Theta(k^\varepsilon))$, $q_r = k^{2r} + r \cdot k^{2r-1}$, $t_r = r \cdot k^{2r-1}$, and $\ell_r = k^{2r-2+\varepsilon}$ (guaranteed to exist by Lemma 9.3.1 as $k = m^{\Omega(1/r)} = \omega(r^{2/\varepsilon})$ by the assumption that $r = o\left(\frac{\log m}{\log \log m}\right)$).
2. Arbitrary group the players into k^2 groups $\mathcal{N} = (N_1, \ldots, N_{k^2})$, whereby each group contains exactly $n_{r-1} = k^{2r-2}$ players.
3. Pick an index $j^* \in [p]$ uniformly at random and sample an instance $I_r^\star \sim \mathcal{D}_{r-1}([n_{r-1}], S_{j^*})$.
4. For each group $N_g \in \mathcal{N}$ independently,
   (a) Define $I_{N_g}^\star$ as $I_r^\star$ by mapping the players in $[n_{r-1}]$ to N_g.
   (b) For each player $i \in N_g$ *independently*, create p instances $I^{(i)} := (I_{i,1}, \ldots, I_{i,p})$ whereby for all $j \neq j^*$, $I_{i,j} \sim \mathcal{D}_{r-1}(N_g, S_j)$, and $I_{i,j^*} = I_{N_g}^\star$.
   (c) For a player $i \in N_g$ and index $j \in [p]$, let $\mathcal{F}_{i,j}$ be the set of *private collection* of that player in instance $I_{i,j}$ and let $\mathcal{F}_i = \bigcup_{j \in [p]} \mathcal{F}_{i,j}$.
5. Pick a random permutation σ of M. For each $g \in [k^2]$ and group N_g, map the k^{2r} items in $[q_r] \setminus S_{j^*}$ to $\sigma(1), \ldots, \sigma(k^{2r})$, and the t_r items in S_{j^*} to $\sigma(k^{2r} + (g-1) \cdot t_r + 1) \ldots \sigma(g \cdot t_r)$ (and for each player $i \in N_g$, update the item set of $\mathcal{F}_i$ and underlying instances $I_{i,1}, \ldots, I_{i,p}$ accordingly).
6. For any player $i \in N$, define the valuation function of player i as $v_i(S) = \max_{T \in \mathcal{F}_i} |S \cap T|$ (these valuation functions are XOS valuation; see Eq (2.2)).

The case of $r = 0$ only includes one player and a set of size N an either the player values this set at N or has no value for the set.

We make several observations about the distribution $\mathcal{D}_r$. Recall that $\mathcal{F}_i$ denotes the private collection of player $i \in N$ that is used to define the valuation function v_i. By construction, the size of the sets inside each private collection is equal across any two

distributions $\mathcal{D}_r$ and $\mathcal{D}_{r'}$ and hence is equal to k (by definition of distribution $\mathcal{D}_1$). A simple property of these sets is that,

Observation 9.4.1. *For any player $i \in N$, and any set $T \in \mathcal{F}_i$, the set T is chosen uniformly at random from all k-subsets of M.*

Fix any group $N_g \in \mathcal{N}$ and any player $i \in N_g$. The input to player i can be seen as the "view" of i in the p instances $I^{(i)} := (I_{i,1}, \ldots, I_{i,p})$, i.e., the input of the i-th player (in N_g) in $I_{i,j}$ (for all $j \in [p]$) and *not* the whole instance. However, in the following, we slightly abuse the notation and use $I_{i,j}$ to also denote the view of player i in the instance $I_{i,j}$. Moreover, we point out that $I_{i,j}$ is defined over the set of items S_j; hence, the complete input to player i is $(I^{(i)}, \phi_i)$ where ϕ_i is the labeling function to map the items S_j to M.

For any player $i \in N$, we refer to the instance I_{i,j^*} of player i as the *special instance* of player i, and to all other instances $I_{i,j}$ for $j \neq j^*$ as *fooling instances*. We further define $I^\star_r(i)$ as the input of player i in the special instance $I_{i,j^*} = I^\star_r$, and define $I^\star_r(-i)$ as the input of all other players in $I^\star_r$.

Observation 9.4.2. *For any group $N_g \in \mathcal{N}$, the joint input of all players $i \in N_g$ in their special instances I_{i,j^*} form the instance $I^\star_{N_g}$ that is sampled from the distribution $\mathcal{D}_{r-1}$.*

On the other hand, the fooling instances of players $i \in N_g$ are sampled *independently* and hence the joint distribution of the players on their instances $I_{i,j}$ is *not* sampled from $\mathcal{D}_{r-1}$. Nevertheless, this difference is *not evident* to the player i.

Observation 9.4.3. *For any player $i \in N$, conditioned on the input $(I^{(i)}, \phi_i)$ given to the player i, the index j^* is chosen uniformly at random from $[p]$.*

Observation 9.4.4. *The distribution of collection of instances $\mathcal{I} := (I^{(1)}, \ldots, I^{(n_r)}) \sim \mathcal{D}_r \mid I^\star_r, \sigma, j^*$ is a* product *distribution as instances in Line (4b) are sampled independently (except for instances $I_{i,j^*} = I^\star_r$ which are already conditioned on above).*

Another important property of the special instances in distribution $\mathcal{D}_r$ is that,

Observation 9.4.5. *The special instances $I^\star_{N_1}, \ldots, I^\star_{N_{k^2}}$ are supported on disjoint set of items (according to the mapping σ).*

Notice that we can trace the special instances into a *unique* path $I^\star_r \to I^\star_{r-1} \to \ldots \to I^\star_2$, whereby $I^\star_2$ is sampled from the distribution $\mathcal{D}_1$. We use $\theta^\star$ to denote the parameter θ (in $\mathcal{D}_1$) in the instance $I^\star_2$ in this path. The following lemma proves a key relation between $\theta^\star$ and social welfare of the sampled instance.

Lemma 9.4.6. *For any instance $I \sim \mathcal{D}_r$:*

$$\Pr\left(\mathsf{sw}(I) \geq k^{2r+1} \mid \theta^\star = 1\right) = 1 \tag{9.1}$$

$$\Pr\left(\mathsf{sw}(I) \leq 2r \cdot k^{2r+2\varepsilon} \mid \theta^\star = 0\right) = 1 - r \cdot \exp\left(-\Omega(k^\varepsilon)\right) \tag{9.2}$$

Proof. We start by the simpler case of Eq (9.1); the proof is by induction. The base case for $r = 0$ is true trivially. Suppose this holds for all integers smaller than r. Now, consider an instance $I \sim (\mathcal{D}_r \mid \theta^\star = 1)$ and the k^2 special instances $I_{N_1}, \ldots, I_{N_{k^2}}$ sampled from $(\mathcal{D}_{r-1} \mid \theta^\star = 1)$ in I. By induction, there is an allocation A_g for each $g \in [k^2]$ that results in a welfare of at least k^{2r-1} in each $I^\star_{N_g}$. By Observation 9.4.5, the set of items among special instances are disjoint, and hence the allocation $A := (A_1, \ldots, A_{k^2})$ which assigns the bundles in A_g to players in N_g for $g \in [k^2]$ is a valid allocation that results in a welfare of $k^2 \cdot k^{2r-1} = k^{2r+1}$, proving the induction step.

We now prove Eq (9.2) by induction. The base case of $r = 0$ is again true trivially. Assume that the bounds hold for all integers smaller than r and consider an instance $I \sim (\mathcal{D}_r \mid \theta^\star = 0)$ and let $I^\star_{N_1}, \ldots, I^\star_{N_{k^2}}$ be the special instances of I, "copied" from the instance $I^\star_r \sim (\mathcal{D}_{r-1} \mid \theta^\star = 0)$ (as in Line (4a) of $\mathcal{D}_r$). Let U be the set of items assigned to these instances (by mapping σ) and $\neg U$ be the set of remaining items assigned by σ, i.e., the items that have no value in the special instances; we have $|U| = k^2 \cdot t_r = r \cdot k^{2r+1}$ and $|\neg U| = k^{2r}$ (notice that σ does *not* assign all the items in M; in particular, $k^{2r+1} - k^{2r}$ items are not assigned to any instance, i.e., have no value for any player; these extra items are only added to simplify the math.). We have,

Claim 9.4.7. *W.p. $1 - \exp\left(-\Omega(k^\varepsilon)\right)$, for any player $i \in N$ and any set $T \in F_i$ such that T does* not *belong to a private collection of a special instance (i.e., T is not sampled from I_{i,j^*}), $|T \cap U| \leq k^{2\varepsilon}$.*

Proof. Fix a group $N_g \in \mathcal{N}$ and fix a player $i \in N_g$ and let $I_{i,j}$ be an instance of $\mathcal{D}_{r-1}$ for some $j \neq j^*$, i.e., not a special instance. Recall that the set of items in $I_{i,j}$ and I_{i,j^*} are two distinct sets S_j and S_{j^*} from $\mathcal{S}_r$ on the universe $[q_r]$ (and hence $|S_j \cap S_{j^*}| \leq \ell_r = k^{2r-2+\varepsilon}$ by definition of intersecting families), and since $[q_r]$ is entirely mapped by σ for player $i \in N_g$, the intersection between item set of $I_{i,j}$ and I_{i,j^*} is at most $k^{2r-2+\varepsilon}$; this in particular means that at most $k^{2r-2+\varepsilon}$ items in $I_{i,j}$ belong to U ($I_{i,j}$ does not share any item with any instance I_{i',j^*} for any $i' \notin N_g$).

Now consider the choice of a set T (in the private collection) for the player i in the instance $I_{i,j}$. For each item a that belongs to both item-set of $I_{i,j}$ and U, define an indicator random variable $X_a \in \{0, 1\}$, which is one iff a is chosen in T. Then, $X := \sum_a X_a$ denotes $|T \cap U|$. By Observation 9.4.1, T is a k-subset chosen uniformly at random from a universe

of size $t_r = r \cdot k^{2r-1}$, and hence, $\mathbb{E}[X] \leq k^{2r-2+\varepsilon} \cdot 1/(r \cdot k^{2r-2}) \leq k^{\varepsilon}/r$. By Chernoff bound for negatively correlated variables (Proposition 7.3.1), $\Pr\left(|S \cap U| \geq k^{2\varepsilon}\right) \leq \exp\left(-\Omega(k^{2\varepsilon})\right)$.

We can now apply a union bound over all possible choices for the set T (among all players and instances), and the probability that even one set T violates this constraint is (note that there are $n_r \cdot p^r$ different choices for T)

$$n_r \cdot p^r \cdot \exp\left(-\Omega(k^{2\varepsilon})\right) = \exp\left(\Theta(r \cdot \log k)\right) \cdot \exp\left(\Theta(r \cdot k^{\varepsilon})\right) \cdot \exp\left(-\Omega(k^{2\varepsilon})\right) = \exp\left(-\Omega(k^{\varepsilon})\right)$$

since $r = o(k^{\varepsilon})$ (by the assumption that $r = o\left(\frac{\log m}{\log\log m}\right)$). □ Claim 9.4.7

In the following we condition on the event in Claim 9.4.7 (event $\mathcal{E}_1$) and the event that $\mathsf{sw}(I^\star) \leq 2(r-1) \cdot k^{2r-2+2\varepsilon}$ (event $\mathcal{E}_2$). Note that by Claim 9.4.7 and induction hypothesis, these two events happen (simultaneously) w.p. $1 - r \cdot \exp\left(-\Omega(k^{\varepsilon})\right)$.

Now fix any allocation $\mathcal{A} = (A_1, \ldots, A_n)$. As size of $\neg U$ is at most k^{2r}, the items in $\neg U$ can only contribute k^{2r} to the welfare in $\mathcal{A}$. Next, let $\mathcal{A}^*$ be the subset of $\mathcal{A}$ such that the maximizing clause in each $A_i \in \mathcal{A}^*$ (i.e., the set $T \in \mathcal{F}_i$) belongs to some special instance, and $\mathcal{A}'$ be the remaining part of allocation $\mathcal{A}$. We know, by $\mathcal{E}_2$, that the contribution of $\mathcal{A}^*$ to the welfare is at most $k^2 \cdot 2(r-1) \cdot k^{2r-2+2\varepsilon} = 2(r-1) \cdot k^{2r+2\varepsilon}$ (counting the k^2 special instances). Moreover, by $\mathcal{E}_1$ (in Claim 9.4.7), the contribution of $\mathcal{A}'$ is at most $k^{2r} \cdot k^{2\varepsilon} = k^{2r+2\varepsilon}$. To conclude, we obtain that the social welfare when $\theta^\star = 0$ is at most $k^{2r} + 2(r-1) \cdot k^{2r+2\varepsilon} + k^{2r+2\varepsilon} \leq 2r \cdot k^{2r+2\varepsilon}$ with the desired probability. □ Lemma 9.4.6

Proof of Theorem 9.2. By Lemma 9.4.6, any r-round protocol that outputs a $\left(\frac{1}{r} \cdot m_r^{\frac{1-2\varepsilon}{2r+1}}\right)$-approximation to the social welfare of instances $I \sim \mathcal{D}_r$, w.p. of failure $\delta < 1/4$, is also a $(\delta + o(1))$-error protocol for estimating the parameter $\theta^\star$. We can hence invoke our framework for proving multi-round lower bounds in Section 8.4 to finalize the proof.

It is straightforward to verify that the (p_r, q_r, t_r, ℓ_r)-intersecting family forms a packing function σ_r (according to Definition 8.1) and the labeling functions ϕ_i chosen in $\mathcal{D}_r$ are indeed forming a labeling family (according to Definition 8.2). They are also clearly locally computable (according to Definition 8.3). Lemma 9.4.6 also implies that these functions are γ-preserving for parameter $\gamma \approx r \cdot \exp\left(-\Omega(k^{\varepsilon})\right)$ (according to Definition 8.4). As such, the distributions $\{\mathcal{D}_r\}$ is a γ-hard family. Thus, by Theorem 8.3, we have $\|\pi\| = \Omega(p/r^4) = \exp\left(\Theta(k^{\varepsilon})\right)/r^4 = \exp\left(m^{\Omega(\varepsilon/r)}\right)/r^4 = \exp\left(m^{\Omega(\varepsilon/r)}\right)$, as $k = m^{\Omega(1/r)}$ and $r = o(\frac{\log m}{\log\log m})$. Re-parametrizing ε by $\varepsilon/2$ in finalizes the proof. □ Theorem 9.2

Chapter 10
Learning With Limited Rounds of Adaptivity

In this last technical chapter of our thesis, we study the power and limitations of adaptivity in learning scenarios. The materials in this chapter are based on [4].

In many learning settings, active/adaptive querying is possible, but the number of rounds of adaptivity—the number of rounds of interaction with the feedback generation mechanism—is limited. For example, in crowdsourcing, one can actively request feedback by sending queries to the crowd, but there is typically a waiting time before queries are answered; if the overall task is to be completed within a certain time frame, this effectively limits the number of rounds of interaction. Similarly, in marketing applications, one can actively request feedback by sending surveys to customers, but there is typically a waiting time before survey responses are received; again, if the marketing campaign is to be completed within a certain time frame, this effectively limits the number of rounds of interaction.

We study the relationship between query complexity and adaptivity in identifying the k most biased coins among a set of n coins with unknown biases. This problem is a common abstraction of many well-studied problems, including the problem of identifying the k best arms in a stochastic multi-armed bandit, and the problem of top-k ranking from pairwise comparisons. Our main result establishes an *optimal* lower bound on the number of rounds adaptivity needed to achieve the optimal worst case query complexity for all these problems. In particular, we show that, perhaps surprisingly, no constant number of rounds suffices for this task, and the "correct" number of rounds of adaptivity is $\log^*(n)$ (an upper bound of $\log^*(n)$ rounds is also established in our paper [4]).

Highlights of Our Contributions

In this chapter, we will establish:

- A tight tradeoff between the degree of adaptivity and the query complexity of algorithms for finding the k-most biased coins (Section 10.4).

Our results also extend to the problems of finding k best arms in a stochastic multi-armed bandit or top-k ranking from pairwise comparisons.

10.1. Background

In the classical *probably approximately correct* (PAC) model [293], the learner is a passive observer who is given a collection of randomly sampled observations from which to learn. In recent years, there has been growing interest in *active learning* models, where the learner can actively request labels or feedback at specific data points; the hope is that, by adaptively

guiding the data collection process, learning can be accomplished with fewer observations than in the passive case. Most learning algorithms operate in one of these settings: learning is either fully passive, or fully active.

We study active/adaptive learning with *limited rounds of adaptivity*, where the learner can actively request feedback at specific data points, but can do so in only a small number of rounds. Specifically, the learner is free to query any number of data points in each round; however, all data points to be queried in a given round must be submitted *simultaneously*, based only on feedback received in previous rounds. In this setting, we are interested not only in bounding the overall query complexity of the learner, but rather in understanding the tradeoff between the number of rounds and the overall query complexity: how many queries are needed given a fixed number of rounds, and conversely, given a target number of queries, how many rounds are necessary?

We study this question in the context of an abstract coin tossing problem, and discuss how the results give us novel insights into the round vs. query complexity tradeoff for two problems that have received increasing interest in the learning theory community in recent years: multi-armed bandits, and ranking from pairwise comparisons[1].

The abstract coin problem we study can be described as follows: say we are given n coins with unknown biases, each of which can be 'queried' by tossing the coin and observing the outcome of the toss. The goal is to find the k coins with highest biases. This problem is a special case of the problem of finding the k best arms in a stochastic multi-armed bandit (MAB), and has received considerable attention in recent years [132, 199, 37, 200, 150, 192, 79, 214, 95, 216, 196, 96]. In particular, it is known that $O\big(\frac{n \log k}{\Delta_k^2}\big)$ coin tosses suffice to find the k most biased coins with arbitrarily high constant probability, where Δ_k is the gap between the k-th and $(k+1)$-th largest biases [199, 132]. It is also known that this bound is optimal in terms of the worst-case query complexity [200, 240]. However, the previous best algorithms for this problem all required $\Omega(\log n)$ rounds of adaptivity to achieve the optimal worst-case query complexity. But are $\Omega(\log n)$ rounds necessary for achieving this optimal query complexity?

In our work in [4], we addressed this question by presenting an algorithm that significantly improved upon the round complexity of state-of-the-art algorithms, yet still achieved the optimal worst-case query complexity: given the gap parameter Δ_k, the algorithm returns the k most biased coins using $O\big(\frac{n \log k}{\Delta_k^2}\big)$ coin tosses with arbitrarily large constant probability in only $\log^*(n)$ rounds of adaptivity.

Considering our results in [4], the natural question at this point is that whether we

[1] In the MAB and ranking literature, the query complexity of an algorithm is often referred to as simply its *sample complexity*. In this paper we use the two terms interchangeably.

can further improve upon the round complexity of this algorithm, and perhaps achieve an $O(1)$ round algorithm with the optimal worst-case query complexity? In this chapter, we refute this possibility by proving that the $\log^*(n)$ bound achieved by our algorithm in [4] is essentially the "right" number of rounds of adaptivity required for obtaining the optimal worst-case query complexity, even when k is only a constant.

10.2. Our Results and Techniques

Our main result is a tight round vs query complexity tradeoff for the coin-tossing problem.

Result 10.1. *For any integer $r \geq 1$, any r-round algorithm that finds the k most biased coins in a set of n coins w.p. at least $3/4$ has query complexity $\Omega\left(\frac{n}{\Delta^2 \cdot r^4} \cdot \text{ilog}^{(r)}(\frac{n}{k})\right)$. Here Δ is the gap between the k-th and $(k+1)$-th largest biases and $\text{ilog}^{(r)}(\cdot)$ denotes the iterated logarithm of order r.*

Our Result 10.1 combined with the algorithm of [4] provides a near complete understanding of the power of additional adaptivity rounds on reducing the query complexity of algorithms for finding top-k most biased coins.

An important corollary of Result 10.1 is that achieving the optimal worst-case query complexity requires (slightly) *super-constant* round complexity:

Corollary 10.2. *Any algorithm that finds the k most biased coins among n coins w.p. at least $3/4$ using the optimal query complexity of $O(n/\Delta^2)$ requires $(\log^*(n) - \log^*(\Theta(\log^* n)))$ rounds. Here Δ is the gap between the k-th and $(k+1)$-th largest biases*

Our round-complexity bound in Corollary 10.2 matches the round-complexity of the algorithm of [4] up to an extremely small *additive* factor of $\log^*(\Theta(\log^* n))$ when k is a constant, implying that $\log^* n$ is indeed "right" number of rounds of adaptivity required for obtaining the optimal worst-case query complexity, even when k is only a constant.

In addition to finding the k best arms in a stochastic multi-armed bandit (MAB), our results for the above coin-tossing problem are also applicable to the problem of top-k ranking from pairwise comparisons (see [4] for the reduction between two problems), another problem that has received considerable interest in recent years [137, 84, 97, 287, 193, 177, 114, 70]. Most top-k ranking approaches we are aware of assume either a non-adaptive setting or a fully adaptive setting; the main exceptions to this are [137, 114, 70], who considered the top-k ranking problem under limited rounds of adaptivity, but under the restricted *noisy permutation* model of pairwise comparisons. Here, we make no assumptions on the underlying pairwise comparison model. Our results for the abstract coin problem imply that $\log^* n$ rounds of adaptivity are required for achieving optimal number of comparisons in top-k ranking from pairwise comparisons, matching the upper bound of [4].

Our techniques. Our proof is based on analyzing a family of "hard" instances for the problem which consists of k heavy coins and $n-k$ light coins. Using information-theoretic machinery, we show that any algorithm that uses small number of coin tosses in the first round can only "trap" the heavy coins in a large pool of candidates. We then inductively show that this forces the algorithm to still solve a "hard" problem on a large domain in the subsequent rounds which we show is not possible due the limited budget of the algorithm.

10.3. Preliminaries

Notation. For a (multi-)set of numbers $\{a_1, \ldots, a_n\}$, we define $a_{[i]}$ as the i-th largest value in this set (ties are broken arbitrarily). For any integer $r \geq 0$, $\text{ilog}^{(r)}(a)$ denotes the iterated logarithms of order r, i.e. $\text{ilog}^{(r)}(a) = \max\left\{\log\left(\text{ilog}^{(r-1)}(a)\right), 1\right\}$ and $\text{ilog}^{(0)}(a) = a$. For any $p \in [0,1]$, $\mathcal{B}(p)$ denotes the Bernoulli distribution with mean p.

Useful information-theory tools. Recall the definition of KL-divergence from Section 2.6.2. We have the following property of KL-divergence (see, e.g., [158], Theorem 5).

Fact 10.3.1. *For any two parameters $0 < p, q < 1$, $\mathbb{D}(\mathcal{B}(p) \mid\mid \mathcal{B}(q)) \leq \frac{(p-q)^2}{q \cdot (1-q)}$.*

We also use the following auxiliary lemma that allows us to decompose the distribution of any random variable with high entropy to a convex combination of a small number of near uniform distributions plus a low probability "noise term".

Lemma 10.3.2 ([33, 4])**.** *Let $\mathsf{A} \sim \mathcal{D}$ be a random variable on $[n]$ with $\mathbb{H}(\mathsf{A}) \geq \log n - \gamma$ for some $\gamma \geq 1$. For any $\varepsilon > \exp(-\gamma)$, there exists $\ell + 1$ distributions $\psi_0, \psi_1, \ldots, \psi_\ell$ on $[n]$ along with $\ell + 1$ probabilities $p_0, p_1, \ldots, p_\ell$ ($\sum_i p_i = 1$) for some $\ell = O(\gamma/\varepsilon^3)$ such that $\mathcal{D} = \sum_{i=1}^{\ell} p_i \cdot \psi_i$, $p_0 = O(\varepsilon)$, and for any $i \geq 1$,*

1. *$\log |\text{SUPP}(\psi_i)| \geq \log n - \gamma/\varepsilon$.*
2. *$|\psi_i - \mathcal{U}_i|_{tvd} = O(\varepsilon)$ where $\mathcal{U}_i$ denotes the uniform distribution on $\text{SUPP}(\psi_i)$.*

Coin-tossing problem. The specific problem we consider can be stated formally as follows: given n coins with unknown biases $p_1, \ldots, p_n$, and an integer $k \in [n]$, the goal is to identify (via tosses of the n coins) the set of k most biased coins. An important parameter in determining the query complexity of this problem is the *gap parameter* $\Delta_k := p_{[k]} - p_{[k+1]}$, i.e. the gap between the k-th and $(k+1)$-th highest biases (recall that $p_{[i]}$ denotes the bias of the i-th most biased coin). We also define $\Delta_i = \max\{\left|p_{[i]} - p_{[k+1]}\right|, \left|p_{[i]} - p_{[k]}\right|\}$. For our lower bound, we will also assume our algorithm is given a lower bound Δ on the gap parameter ($\Delta_k \geq \Delta > 0$).[2] This can clearly only strengthens our lower bound.

[2]The assumption about knowledge of Δ is also common in the MAB and ranking literature; see, e.g., [132, 199, 97, 287].

We are interested here in algorithms that require limited *rounds* of adaptivity. In each round, an algorithm can decide to query various coins by tossing them (with no limit on the number of coins that can be tossed in a round or on the number of times any given coin can be tossed in a round); however, all tosses to be conducted in a given round must be chosen *simultaneously*, based only on the outcomes observed in previous rounds. We say an algorithm is an *r-round algorithm* if it uses at most r rounds of adaptivity; the total number of coin tosses it uses is termed its *query complexity*. For any $\delta \in [0, 1)$, we say an algorithm is a *δ-error algorithm* for the above problem if it correctly returns the set of k most biased coins with probability at least $1 - \delta$.

10.4. A Tight Round-Query Tradeoff for Coin-tossing

In this section, we formalize Result 10.1 by proving the following theorem.

Theorem 10.3. *For any parameter $\Delta \in (0, \frac{1}{2})$ and any integers $n, k \geq 1$, there exists a distribution $\mathcal{D}$ on instances of the k most biased coins problem with n coins and gap parameter $\Delta_k = \Delta$ such that for any integer $r \geq 1$, any r-round algorithm that finds the k most biased coins in instances sampled from $\mathcal{D}$ w.p. at least $3/4$ has query complexity $\Omega\left(\frac{n}{\Delta^2 \cdot r^4} \cdot \text{ilog}^{(r)}(\frac{n}{k})\right)$.*

We first prove Theorem 10.3 for the case of $k = 1$, i.e., the case of finding the most biased coin and then show that a simple reduction extends this result to all possible values of k. Throughout this section, for any algorithm $\mathcal{A}$, $\mathsf{cost}(\mathcal{A})$ denotes the query complexity of $\mathcal{A}$ and $\mathsf{deg}(\mathcal{A})$ denotes the degree of adaptivity it uses, i.e., its round complexity.

Overview. Consider the following input for the best coins problem introduced earlier. we have a collection of $n - 1$ *light* coins and single *heavy* coin with the difference of Δ between the bias of the heavy coin and any light coin. A textbook result is that to classify a single coin as heavy or light correctly with sufficiently large constant probability $\Omega(1/\Delta^2)$ coins tosses are needed (see, e.g., [98]). Using this, it is possible to argue that $\Omega(n/\Delta^2)$ coin tosses are needed in these instances to recover the heavy coin. However notice that Theorem 10.3 is proving a stronger result on the query complexity of r-round algorithms for any $r \leq \log^* n - \log^* \Theta(\log^* n)$. To achieve this stronger bound, we take on a different approach as described below. For the purpose of the following discussion, it would be convenient to see Δ as some constant and hence suppress the bounds on Δ in asymptotic notation. We emphasize that this assumption is only for the purpose of following discussion.

Our starting point is the following key claim that we prove: if an algorithm only tosses $n \cdot s$ coins in the first round, then it can only reduce the set of candidate coins to $n - 2^{\Theta(s)}$ possible coins. More formally, conditioned on the outcome of the first $n \cdot s$ coin tosses, the heavy coin is still distributed (almost) uniformly over a set of $n - 2^{\Theta(s)}$ possible coins.

Having this result, it is then easy to argue that one needs to set $s = \Omega(\log n)$ to recover the heavy coin in one round, resulting in an $\Omega(n \log n)$ lower bound on the query complexity of 1-round algorithms. There is also a more important takeaway from this discussion: any r-round algorithm that does not spend relatively large number of coin tosses in its first round is forced to find the heavy coin from a large pool of candidates (with essentially no further information) in the next $(r-1)$ rounds. Consequently, we can prove Theorem 10.3 inductively, by showing that if the number of coin tosses of an r-round algorithm is $o(n \cdot \text{ilog}^{(r-1)}(n))$, then by setting $s = o(\text{ilog}^{(r)}(n))$ in the above argument, we end up with $\approx n/\sqrt{\text{ilog}^{(r-1)}(n)}$ possible choices for the heavy coin after the first round that needs to be further pruned in the subsequent $(r-1)$ rounds. But by induction, we need ,

$$\Omega(\left(n/\sqrt{\text{ilog}^{(r-1)}(n)}\right) \cdot \text{ilog}^{(r-1)}(n) = \Omega(n \cdot \sqrt{\text{ilog}^{(r-1)}(n)}) = \omega(n \cdot \text{ilog}^{(r)}(n))$$

many coin tosses to solve the problem in $(r-1)$ rounds (over $n/\sqrt{\text{ilog}^{(r-1)}(n)}$ coins), a contradiction with the bounds on the query complexity of the r-round algorithm.

To make the latter intuition precise we use a "round elimination" argument (in spirit of round eliminations in Chapters 8 and 9). We show that given any "good" r-round algorithm (i.e., a one with better query complexity than the bound in Theorem 10.3), there should exists a set of observed coins tosses outcome in the first round, such that conditioned on these coin tosses two events simultaneously happen: (i) the algorithm still outputs a correct answer with essentially the same probability even after this conditioning, and (ii), the distribution of the heavy coin is close to uniform (in total variation distance) on a "large" subset of coins. Having this, we create a "good" $(r-1)$-round algorithm (defined as before) which "embed" its set of coins in the support of the distribution for heavy coin in above discussion and simulates the "missing input" (on the larger domain) for the r-round algorithm using independent randomness, which contradicts the induction step.

We now formalize the proof outlined above. Fix any arbitrary value $\Delta \in (0, 1/2)$ (possibly a function of n) and a constant $p < 1 - \Delta$.

Distribution $\mathcal{D}_n^{\Delta,p}$. A hard input distribution on n coins for finding the most biased coin with the gap parameter $\Delta_1 = \Delta$.

- Sample an index $i^* \in [n]$ uniformly at random.
- Let $p_{i^*} = p + \Delta$ and $p_i = p$ for any $i \neq i^*$ in $[n]$.
- Return the coins $[n]$ with biases $\{p_i\}_{i=1}^n$.

It is immediate to see that in any instance sampled from $\mathcal{D}_n^{\Delta,p}$, $\Delta_1 = \Delta$. Moreover, one

can see that finding the most biased coin in this family of instances is equivalent to finding i^*. We use this fact to prove the lower bound.

Define the recursive function $e(r) = e(r-1) + o(1/r^2)$ with $e(1) = 0$.

Lemma 10.4.1. *Fix any integers $n, r \geq 1$. Suppose $\mathcal{A}_r$ is an r-round algorithm that given an instance sampled from $\mathcal{D}_n^{\Delta,p}$ outputs the most biased coin correctly w.p. at least $2/3+e(r)$; then, $\mathsf{cost}(\mathcal{A}_r) = \Omega(\frac{n}{\Delta^2 \cdot r^4} \cdot \mathrm{ilog}^{(r)}(n))$.*

Fix $n, r \geq 1$ and algorithm $\mathcal{A}_r$ as in Lemma 10.4.1. Note that by an averaging argument, we can assume w.l.o.g that $\mathcal{A}_r$ is deterministic (similar to the easy direction of Yao's minimax principle in Proposition 2.7.2). Indeed, for any randomized algorithm that errs w.p. at most δ on the distribution $\mathcal{D}_n^{\Delta,p}$, there exists a choice of random bits (used by the algorithm) that conditioned on, the error probability of algorithm is still at most δ where the probability is taken over the randomness of the distribution and observed outcomes. Hence, by conditioning on these random bits we obtain a deterministic algorithm with the same performance guarantee. Consequently, we assume that the algorithm $\mathcal{A}_r$ is deterministic.

To continue, we need some notation. Recall that S_1 denotes the multi-set of coins tossed in the first round by $\mathcal{A}_r$. Let s_1 denote the size of S_1 counting the multiplicities. We define the *outcome profile* of S_1 as the s_1-dimensional tuple $T = ((i_1, \theta_1), (i_2, \theta_2), \ldots, (i_{s_1}, \theta_{s_1}))$, whereby for any $j \in [s_1]$, i_j and θ_j, denote, respectively, the index of the j-th coin in S_1 and its value, i.e., heads or tails. We use I to denote the random variable for the index i^* in $\mathcal{D}_n^{\Delta,p}$, and T to denote the random variable for the vector T. We further use Θ_j, for any $j \in [s_1]$, to denote the random variable for parameter θ_j defined above. We let $\Theta := (\Theta_1, \ldots, \Theta_{s_1})$.

Notice that random variables $\Theta_1, \ldots, \Theta_{s_1}$ are in general correlated in the distribution $\mathcal{D}_n^{\Delta,p}$. However, we argue that for any $j \in [s_1]$, Θ_j and Θ^{-j} are *independent conditioned on* I. Indeed, conditioning on I fixes the distribution of the coins: Bernoulli $\mathcal{B}(p+\Delta)$ for the coin i^* and Bernoulli $\mathcal{B}(p)$ for the remaining coins. Therefore, since for all $j \in [s_1]$, Θ_j is sampled from the distribution of the coin i_j, we have, $\Theta_j \perp \Theta^{-j} \mid \mathsf{I}$.

The following lemma bounds the "information" revealed about the index i^* (i.e., the most biased coin) in *the first round* based on coin tosses done by $\mathcal{A}_r$ in this round.

Lemma 10.4.2. $\mathbb{I}(\mathsf{I}\,;\mathsf{T}) = O(s_1 \cdot \Delta^2/n)$.

Proof. Recall that $\mathcal{A}_r$ is a deterministic algorithm. This means that the multi-set S_1 of the coins being tossed in the first round by $\mathcal{A}_r$ is fixed a-priori. As such, the random variable

T is only a function of the vector $\Theta = (\Theta_1, \ldots, \Theta_{s_1})$. We have,

$$\mathbb{I}(\mathsf{I}\,;\mathsf{T}) = \mathbb{I}(\mathsf{I}\,;\Theta) \underset{\text{Fact 2.6.1-(6)}}{=} \sum_{j=1}^{s_1} \mathbb{I}(\mathsf{I}\,;\Theta_j \mid \Theta^{<j}) \underset{\text{Proposition 2.6.4}}{\leq} \sum_{j=1}^{s_1} \mathbb{I}(\mathsf{I}\,;\Theta_j) \tag{10.1}$$

where the inequality is true since $\Theta_j \perp \Theta^{<j} \mid \mathsf{I}$ and hence conditioning on $\Theta^{<j}$ can only decrease the mutual information by Proposition 2.6.4. We now bound each term $\mathbb{I}(\mathsf{I}\,;\Theta_j)$ in the above sum. In order to do this, we write,

$$\mathbb{I}(\mathsf{I}\,;\Theta_j) = \mathop{\mathbb{E}}_{i^* \sim \mathcal{U}([n])} [\mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} = i^*))] \tag{10.2}$$

as i^* is chosen uniformly at random from $[n]$ in $\mathcal{D}_n^{\Delta,p}$. Here $\mathcal{U}([n])$ denotes the uniform distribution on $[n]$ and $\mathbb{D}$ denotes the KL-divergence; see Section 2.6.2 for the definition and Fact 2.6.6 for its connection to mutual information that leads to Eq (10.2). The following claim bounds each term above individually.

Claim 10.4.3. *For any $j \in [s_1]$, $\mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} = i^*)) = \begin{cases} O(\Delta^2) & \text{if } i_j = i^* \\ O(\Delta^2/n^2) & \text{otherwise} \end{cases}$.*

Proof. Fix any $j \in [s_1]$. By definition of $\mathcal{D}_n^{\Delta,p}$, we have,

$$\textsc{dist}(\Theta_j) = \mathcal{B}(p + \Delta/n),\ \textsc{dist}(\Theta_j \mid \mathsf{I} = i_j) = \mathcal{B}(p + \Delta),\ \text{and } \textsc{dist}(\Theta_j \mid \mathsf{I} \neq i_j) = \mathcal{B}(p).$$

Suppose first $i_j = i^*$. In this case,

$$\begin{aligned}\mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} = i^*)) &= \mathbb{D}(\mathcal{B}(p + \Delta/n) \mid\mid \mathcal{B}(p + \Delta)) \\ &\underset{\text{Fact 10.3.1}}{\leq} \frac{(p + \Delta/n - (p + \Delta))^2}{(p + \Delta) \cdot (1 - (p + \Delta))} = O(\Delta^2),\end{aligned}$$

since $p + \Delta$ and $(1 - (p + \Delta))$ are both constants larger than zero. Similarly, if $i_j \neq i^*$,

$$\begin{aligned}\mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} = i^*)) &= \mathbb{D}(\mathcal{B}(p + \Delta/n) \mid\mid \mathcal{B}(p)) \\ &\underset{\text{Fact 10.3.1}}{\leq} \frac{(p + \Delta/n - p)^2}{p \cdot (1 - p)} = O(\Delta^2/n^2).\end{aligned}$$

□ Claim 10.4.3

By plugging in the bounds established in Claim 10.4.3 to Eq (10.2), we have,

$$\mathbb{I}(\mathsf{I}\,;\Theta_j) = \frac{1}{n} \cdot \mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} = i_j)) + \frac{n-1}{n} \cdot \mathbb{D}(\textsc{dist}(\Theta_j) \mid\mid \textsc{dist}(\Theta_j \mid \mathsf{I} \neq i_j))$$

$$\underset{\text{Claim 10.4.3}}{=} O(\Delta^2/n) + O(\Delta^2/n^2) = O(\Delta^2/n).$$

Plugging in this in Eq (10.1) implies $\mathbb{I}(\mathsf{I}\,;\mathsf{T}) \leq \sum_{j=1}^{s_1} O(\Delta^2/n) = O(s_1 \cdot \Delta^2/n)$. $\square$ Lemma 10.4.2

We are now ready to prove Lemma 10.4.1. The proof is by induction (on the number of rounds r). In the following claim, we prove the base case of this induction.

Claim 10.4.4. *With the assumption of Lemma 10.4.1,* $\mathsf{cost}(\mathcal{A}_1) = \Omega(\frac{n}{\Delta^2} \cdot \log n)$.

Proof. We use Fano's inequality (Fact 2.6.2) to prove this claim. Let $\delta = 1/3 - e(1) = 1/3$. Recall that algorithm $\mathcal{A}_1$ tosses the coins S_1 and given the value of these coin tosses in the outcome profile T determines the most biased coin w.p. at least $1 - \delta = 2/3$. As argued earlier, determining the most biased coin in distribution $\mathcal{D}_n^{\Delta,p}$ is equivalent to determining the value of index i^*. As such, T is an δ-error estimator for I. Hence,

$$\delta \cdot |\mathsf{I}| + H_2(\delta) \underset{\text{Fact 2.6.2}}{\geq} \mathbb{H}(\mathsf{I} \mid \mathsf{T}) = \mathbb{H}(\mathsf{I}) - \mathbb{I}(\mathsf{I}\,;\mathsf{T})$$

Now notice that $|\mathsf{I}| = \mathbb{H}(\mathsf{I}) = \log n$ by Fact 2.6.1-(1) as I is uniform over $[n]$. Moreover, by Lemma 10.4.2, $\mathbb{I}(\mathsf{I}\,;\mathsf{T}) = O(s_1 \cdot \Delta^2/n)$. By reordering the terms above, we obtain that $s_1 = \Omega(\frac{n}{\Delta^2} \cdot \log n)$. Noting that $\mathsf{cost}(\mathcal{A}_1) = s_1$ finalizes the proof. $\square$ Claim 10.4.4

Now assume inductively that Lemma 10.4.1 is true for all integers smaller than r and we want to prove this for r-round algorithms. The proof of induction step is by contradiction. We show that if there exists an algorithm $\mathcal{A}_r$ with smaller query complexity than the bounds stated in Lemma 10.4.1 for $\mathcal{D}_n^{\Delta,p}$, then there also exists an $(r-1)$-round algorithm $\mathcal{A}_{r-1}$ with smaller query complexity on distribution $\mathcal{D}_m^{\Delta,p}$ for some appropriately chosen $m \leq n$, which contradicts the induction hypothesis.

Lemma 10.4.2 essentially implies that if the number of coin tosses in the first round is small, then the outcome profile T, on average, does not reveal much information about the identity of the most biased coin, i.e., I. More formally, if we assume (by way of contradiction) that $\mathsf{cost}(\mathcal{A}_r) = o(\frac{n}{\Delta^2 \cdot r^4} \cdot \mathrm{ilog}^{(r)}(n))$, we have,

$$\mathbb{H}(\mathsf{I} \mid \mathsf{T}) = \mathbb{H}(\mathsf{I}) - \mathbb{I}(\mathsf{I}\,;\mathsf{T}) \underset{\text{Fact 2.6.1-(1)}}{=} \log n - \mathbb{I}(\mathsf{I}\,;\mathsf{T}) \underset{\text{Lemma 10.4.2}}{=} \log n - o(\mathrm{ilog}^{(r)}(n)/r^4), \tag{10.3}$$

where in the last part we used the fact that $s_1 \leq \mathsf{cost}(\mathcal{A}_r)$. Now consider any fixed possible outcome profile T for $\mathcal{A}_r$ in round one, i.e., any possible value for T. We say that T is *uninformative* iff $\mathbb{H}(\mathsf{I} \mid \mathsf{T} = T) = \log n - o(\mathrm{ilog}^{(r)}(n)/r^2)$. Roughly speaking, whenever the outcome profile in the first round is uninformative, the algorithm is quite "uncertain"

about the identity of the most biased coin, and hence needs to find it among a large pool of candidate coins in the next $(r-1)$ rounds. This we argue is not possible as by induction hypothesis the available budget is not large enough to solve the problem in $(r-1)$ rounds on such a large domain (by induction hypothesis).

We start by showing that our assumption on query complexity of $\mathcal{A}_r$ implies that there exists an uninformative outcome profile in the first round which still results in a good output by $\mathcal{A}_r$ in the subsequent rounds.

Claim 10.4.5. *There exists an uninformative outcome profile T_{ui} of coin tosses in the first round of $\mathcal{A}_r$ such that* $\Pr(\mathcal{A}_r \text{ errs} \mid \mathsf{T} = T_{\mathsf{ui}}) \leq \delta + o(1/r^2)$.

Proof. Let $C := \log n - \mathbb{H}(\mathsf{I} \mid \mathsf{T})$. By Eq (10.3), $C = o(\text{ilog}^{(r)}(n)/r^4)$. For any $\varepsilon > 0$,

$$\begin{aligned}\Pr_T\left(\log n - \mathbb{H}(\mathsf{I} \mid \mathsf{T} = T) \geq \frac{r^2}{\varepsilon} \cdot C\right) &\leq \frac{\varepsilon \cdot \mathbb{E}_T\left[\log n - \mathbb{H}(\mathsf{I} \mid \mathsf{T} = T)\right]}{r^2 \cdot C} \\ &= \frac{\varepsilon \cdot (\log n - \mathbb{H}(\mathsf{I} \mid \mathsf{T}))}{r^2 \cdot C} = \frac{\varepsilon}{r^2}. \\ &\qquad \text{(by the choice of } C = \log n - \mathbb{H}(\mathsf{I} \mid \mathsf{T}))\end{aligned}$$

This means that w.p. at least $1 - \varepsilon/r^2$, $\mathbb{H}(\mathsf{I} \mid \mathsf{T} = T) \geq \log n - \frac{1}{\varepsilon} \cdot o(\text{ilog}^{(r)}(n)/r^2)$. By taking ε small enough, we have that the probability that T is uninformative is $1 - o(1/r^2)$. Hence,

$$\Pr(\mathcal{A}_r \text{ errs} \mid T \text{ is uninformative}) \leq \Pr(\mathcal{A}_r \text{ errs}) + \Pr(T \text{ is not uninformative}) \leq \delta + o(1/r^2).$$

The assertion of the claim now follows by an averaging argument. □ Claim 10.4.5

Let T_{ui} be the uninformative profile in Claim 10.4.5 and define $\psi := \text{DIST}(\mathsf{I} \mid \mathsf{T} = T_{\mathsf{ui}})$. As $\mathbb{H}(\mathsf{I} \mid \mathsf{T} = T_{\mathsf{ui}}) = \log n - o(\text{ilog}^{(r)}(n)/r^2)$, we can apply Lemma 10.3.2 on random variable $\mathsf{I} \mid \mathsf{T} = T_{\mathsf{ui}}$ with parameters $\gamma = o(\text{ilog}^{(r)}(n)/r^2)$ and $\varepsilon = o(1/r^2)$ to express its distribution ψ as a convex combination of distributions $\psi_0, \psi_1, \ldots, \psi_k$, i.e., $\psi = \sum_i q_i \cdot \psi_i$ (for $\sum_i q_i = 1$) such that $q_0 = o(1/r^2)$ and for all $i \geq 1$,

$$|\text{SUPP}(\psi_i)| \geq 2^{\left(\log n - o(\text{ilog}^{(r)}(n))\right)} \geq \frac{n}{\left(\text{ilog}^{(r-1)}(n)\right)^{o(1)}}, \tag{10.4}$$

$$|\psi_i - \mathcal{U}_i|_{tvd} = o(1/r^2), \tag{10.5}$$

where $\mathcal{U}_i$ is the uniform distribution on $\text{SUPP}(\psi_i)$. With this notation,

$$\delta + o(1/r^2) \underset{\text{Claim 10.4.5}}{\geq} \Pr(\mathcal{A}_r \text{ errs} \mid \mathsf{T} = T_{\mathsf{ui}}) = \sum_i q_i \cdot \Pr(\mathcal{A}_r \text{ errs} \mid \mathsf{T} = T_{\mathsf{ui}}, \mathsf{I} \sim \psi_i).$$

As $q_0 = o(1/r^2)$, by an averaging argument, we have that there exists a distribution ψ_i for some $i \geq 1$ such that $\Pr\left(\mathcal{A}_r \text{ errs} \mid \mathsf{T} = T_{\mathsf{ui}}, \mathsf{I} \sim \psi_i\right) \leq \delta + o(1/r^2)$. Without loss of generality let this distribution be ψ_1 and define $m := |\text{SUPP}(\psi_1)|$. We now use the fact that ψ_1 is close to a uniform distribution on $\text{SUPP}(\psi_1)$ (in total variation distance) together with an embedding argument to show that,

Claim 10.4.6. *There exists a deterministic $(r-1)$-round $\left(\delta + o(1/r^2)\right)$-error algorithm for the best coins problem on $\mathcal{D}_m^{\Delta,p}$ with query complexity at most equal to $\mathsf{cost}(\mathcal{A}_r)$ on $\mathcal{D}_n^{\Delta,p}$.*

Proof. Let $\mathcal{A}_{r,T_{\mathsf{ui}}}$ be an $(r-1)$-round algorithm obtained by running $\mathcal{A}_r$ from the second round onwards assuming that the outcome profile in the first round was T_{ui}. We use $\mathcal{A}_{r,T_{\mathsf{ui}}}$ to design a randomized algorithm $\mathcal{A}'$ for $\mathcal{D}_m^{\Delta,p}$.

Given any instance sampled from $\mathcal{D}_m^{\Delta,p}$, $\mathcal{A}'$ maps $[m]$ to $\text{SUPP}(\psi_1)$ (using any arbitrary bijection). Next, it runs $\mathcal{A}_{r,T_{\mathsf{ui}}}$ as follows: if $\mathcal{A}_{r,T_{\mathsf{ui}}}$ choose to toss a coin in $\text{SUPP}(\psi_1)$, $\mathcal{A}'$ also choose the corresponding coin in $[m]$; otherwise, if $\mathcal{A}_{r,T_{\mathsf{ui}}}$ choose to toss a coin in $[n] \setminus \text{SUPP}(\psi_1)$, $\mathcal{A}'$ simply toss a coin from the distribution $\mathcal{B}(p)$ and return the result to $\mathcal{A}_{r,T_{\mathsf{ui}}}$. Finally, if $\mathcal{A}_{r,T_{\mathsf{ui}}}$ outputs a coin from $\text{SUPP}(\psi_1)$, $\mathcal{A}'$ returns the corresponding coin in $[m]$ and otherwise $\mathcal{A}'$ simply return an arbitrary coin in $[m]$ as the answer.

It is trivially true that $\mathsf{cost}(\mathcal{A}') \leq \mathsf{cost}(\mathcal{A})$. Hence, in the following we prove the correctness of $\mathcal{A}'$. Let $\mathcal{D}'$ be the distribution of underlying instances on $[n]$ created by $\mathcal{A}'$. Let $\mathcal{U}_1$ be the uniform distribution on $\text{SUPP}(\psi_1)$. It is straightforward to verify that $\mathcal{D}' = \mathcal{D}_n^{\Delta,p} \mid \mathsf{I} \sim \mathcal{U}_1$, and that $\mathcal{D}'$ is a deterministic function of I. As such,

$$\begin{aligned}
\Pr_{\mathcal{D}_m^{\Delta,p}}\left(\mathcal{A}' \text{ errs}\right) &= \Pr_{\mathcal{D}'}\left(\mathcal{A}_{r,T_{\mathsf{ui}}} \text{ errs}\right) = \Pr_{\mathcal{D}_n^{\Delta,p}}\left(\mathcal{A}_{r,T_{\mathsf{ui}}} \text{ errs} \mid \mathsf{I} \sim \mathcal{U}_1\right) \\
&\underset{\text{Fact 2.6.7}}{\leq} \Pr_{\mathcal{D}_n^{\Delta,p}}\left(\mathcal{A}_{r,T_{\mathsf{ui}}} \text{ errs} \mid \mathsf{I} \sim \psi_1\right) + |\psi_1 - \mathcal{U}_1|_{tvd} \\
&\underset{\text{Eq (10.5)}}{=} \Pr_{\mathcal{D}_n^{\Delta,p}}\left(\mathcal{A}_r \text{ errs} \mid \mathsf{I} \sim \psi_1, \mathsf{T} = T_{\mathsf{ui}}\right) + o(1/r^2) \\
&\leq \delta + o(1/r^2).
\end{aligned}$$

To finalize the proof, note that by an averaging argument, there exists a fixing of the randomness in $\mathcal{A}'$ that results in the same error guarantee. This results in a deterministic $(r-1)$-round algorithm $\mathcal{A}''$ that errs on $\mathcal{D}_m^{\Delta,p}$ w.p. at most $\delta + o(1/r^2)$. □ Claim 10.4.6

We are now ready to conclude the proof of Lemma 10.4.1. By Claim 10.4.6, there exists an $(r-1)$-round algorithm $\mathcal{A}_{r-1}$ that errs w.p. at most $\delta + o(1/r^2) = 1/3 - e(r) + o(1/r^2) = 1/3 - e(r-1)$ on instances of $\mathcal{D}_m^{\Delta,p}$ such that $\mathsf{cost}(\mathcal{A}_{r-1}) \leq \mathsf{cost}(\mathcal{A}_r)$. But by induction

hypothesis (as $\mathcal{A}_{r-1}$ is an $(r-1)$-round algorithm), we know,

$$\begin{aligned}
\mathsf{cost}(\mathcal{A}_{r-1}) &= \Omega\left(\frac{m}{\Delta^2\cdot(r-1)^4}\cdot \mathrm{ilog}^{(r-1)}(m)\right)\\
&\underset{\text{Eq (10.4)}}{=} \Omega\left(\frac{1}{\Delta^2\cdot(r-1)^4}\cdot\frac{n}{\left(\mathrm{ilog}^{(r-1)}(n)\right)^{o(1)}}\cdot \mathrm{ilog}^{(r-1)}(n)\right)\\
&= \Omega\left(\frac{n}{\Delta^2\cdot r^4}\cdot\left(\mathrm{ilog}^{(r-1)}(n)\right)^{1-o(1)}\right)\\
&= \Omega\left(\frac{n}{\Delta^2\cdot r^4}\cdot \mathrm{ilog}^{(r)}(n)\right), \qquad (\text{as } \mathrm{ilog}^{(r)}(n)=\log(\mathrm{ilog}^{(r-1)}(n)))
\end{aligned}$$

which is in contradiction with $\mathsf{cost}(\mathcal{A}_{r-1}) \le \mathsf{cost}(\mathcal{A}_r) = o\left(\frac{n}{\Delta^2\cdot r^4}\cdot \mathrm{ilog}^{(r)}(n)\right)$. This finalizes the proof of Lemma 10.4.1.

We can now conclude the proof of Theorem 10.3.

Proof of Theorem 10.3. Fix parameters Δ and integers n, k. For simplicity, we assume k divides n. We further pick a constant $p < 1-\Delta$. Create distribution $\mathcal{D}$ as follows:

1. Partition the set $[n]$ of coins into k equal size subsets $N_1, \ldots, N_k$ each of size $t := n/k$.
2. For any $j \in [k]$, we sample the bias of the coins in N_j from $\mathcal{D}_t^{\Delta,p}$.

Notice that in any instance sampled from distribution $\mathcal{D}$, there are k coins with bias $p+\Delta$ and $n-k$ coins with bias p, and hence $\Delta_k = \Delta$. Additionally, each of the coins with bias $p+\Delta$ belongs to a separate subset N_j. Hence, finding the top k most biased coins in distribution $\mathcal{D}$ amounts to finding the most biased coins in k *independent* instances sampled from $\mathcal{D}_t^{\Delta,p}$. We now use this to prove the lower bound.

Let $\mathcal{A}$ be a $(1/4)$-error r-round algorithm for finding the top k most biased coins in $\mathcal{D}$, and assume by contradiction that $\mathsf{cost}(\mathcal{A}) = o\left(\frac{n}{\Delta^2}\cdot \mathrm{ilog}^{(r)}(\frac{n}{k})\right)$. This means that there exists at least one index $j \in [k]$, such that in expectation, only $o(\frac{n}{k\cdot\Delta^2}\cdot \mathrm{ilog}^{(r)}(\frac{n}{k}))$ coins are being tossed in the coins in N_j. By Markov inequality, we have that w.p. $1-o(1)$, only $o(\frac{n}{k\cdot\Delta^2}\cdot \mathrm{ilog}^{(r)}(\frac{n}{k}))$ coins are being tossed in N_j; for brevity, let $\mathcal{E}$ denote this event. We have, $\Pr(\mathcal{A} \text{ finds the most biased coin in } N_j \mid \mathcal{E}) \ge 3/4 - o(1)$.

This means that $\mathcal{A}$ when restricted to the coins in N_j, finds the top most biased coin w.p. at least $3/4-o(1)$ using at most $o(\frac{t}{\Delta^2}\cdot \mathrm{ilog}^{(r)}(t))$ many coin tosses (recall that $t = n/k$). On the other hand, by Lemma 10.4.1, any algorithm for finding the most biased coins in instances sampled from $\mathcal{D}_t^{\Delta,p}$ w.p. at least $2/3 + e(r) = 2/3 + \sum_{i=1}^{r} o(1/i^2) = 2/3 + o(1) < 3/4 - o(1)$, requires $\Omega(\frac{t}{\Delta^2}\cdot \mathrm{ilog}^{(r)}(t))$ many coin tosses, a contradiction. □ Theorem 10.3

Bibilography

[1] F. M. Ablayev. Lower bounds for one-way probabilistic communication complexity. In *Automata, Languages and Programming, 20nd International Colloquium, ICALP93, Lund, Sweden, July 5-9, 1993, Proceedings*, pages 241–252, 1993. 40

[2] F. N. Afrati, V. Borkar, M. Carey, N. Polyzotis, and J. D. Ullman. Map-reduce extensions and recursive queries. In *Proceedings of the 14th international conference on extending database technology*, pages 1–8. ACM, 2011. 137

[3] F. N. Afrati, A. D. Sarma, S. Salihoglu, and J. D. Ullman. Upper and lower bounds on the cost of a map-reduce computation. *PVLDB*, 6(4):277–288, 2013. 15, 211

[4] A. Agarwal, S. Agarwal, S. Assadi, and S. Khanna. Learning with limited rounds of adaptivity: Coin tossing, multi-armed bandits, and ranking from pairwise comparisons. In *Proceedings of the 30th Conference on Learning Theory, COLT 2017, Amsterdam, The Netherlands, 7-10 July 2017*, pages 39–75, 2017. 24, 256, 257, 258, 259

[5] P. K. Agarwal, G. Cormode, Z. Huang, J. M. Phillips, Z. Wei, and K. Yi. Mergeable summaries. *ACM Trans. Database Syst.*, 38(4):26, 2013. 10

[6] R. Aharoni, P. Erdös, and N. Linial. Optima of dual integer linear programs. *Combinatorica*, 8(1):13–20, 1988. 249

[7] K. J. Ahn and S. Guha. Graph sparsification in the semi-streaming model. In *Automata, Languages and Programming, 36th Internatilonal Colloquium, ICALP 2009, Rhodes, Greece, July 5-12, 2009, Proceedings, Part II*, pages 328–338, 2009. 12

[8] K. J. Ahn and S. Guha. Linear programming in the semi-streaming model with application to the maximum matching problem. *Inf. Comput.*, 222:59–79, 2013. 51, 56

[9] K. J. Ahn and S. Guha. Access to data and number of iterations: Dual primal algorithms for maximum matching under resource constraints. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 202–211, 2015. 14, 51, 53, 56, 96, 119, 120, 134, 211

[10] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467. SIAM, 2012. 11, 90, 96, 119, 120, 134, 137, 170, 208, 211

[11] K. J. Ahn, S. Guha, and A. McGregor. Analyzing graph structure via linear measurements. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 459–467. SIAM, 2012. 13, 14, 51, 89

[12] K. J. Ahn, S. Guha, and A. McGregor. Graph sketches: sparsification, spanners, and subgraphs. In *Proceedings of the 31st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2012, Scottsdale, AZ, USA, May 20-24, 2012*, pages 5–14, 2012. 13, 89, 90

[13] K. J. Ahn, S. Guha, and A. McGregor. Spectral sparsification in dynamic graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 16th International Workshop, APPROX 2013, and 17th International Workshop, RANDOM 2013, Berkeley, CA, USA, August 21-23, 2013. Proceedings*, pages 1–10, 2013. 13, 89

[14] Y. Ai, W. Hu, Y. Li, and D. P. Woodruff. New characterizations in turnstile streams with applications. In *31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan*, pages 20:1–20:22, 2016. 11, 44, 45, 54, 65, 70, 77, 96, 210, 243

[15] R. Aleliunas, R. M. Karp, R. J. Lipton, L. Lovász, and C. Rackoff. Random walks, universal traversal sequences, and the complexity of maze problems. In *20th Annual Symposium on Foundations of Computer Science, 29-31 October 1979*, pages 218–223, 1979. 142

[16] N. Alon. Testing subgraphs in large graphs. *Random Struct. Algorithms*, 21(3-4):359–370, 2002. 30

[17] N. Alon, Y. Matias, and M. Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29. ACM, 1996. 5, 43, 44, 177

[18] N. Alon, A. Moitra, and B. Sudakov. Nearly complete graphs decomposable into large induced matchings and their applications. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1079–1090, 2012. 30, 55, 67, 77, 85, 95, 212

[19] N. Alon, N. Nisan, R. Raz, and O. Weinstein. Welfare maximization with limited interaction. In *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1499–1512, 2015. 13, 22, 96, 209, 212, 213, 215, 246, 248, 249

[20] N. Alon and A. Shapira. A characterization of easily testable induced subgraphs. *Combinatorics, Probability & Computing*, 15(6):791–805, 2006. 30

[21] A. Ambainis, H. Buhrman, W. I. Gasarch, B. Kalyanasundaram, and L. Torenvliet. The communication complexity of enumeration, elimination, and selection. *J. Comput. Syst. Sci.*, 63(2):148–185, 2001. 49

[22] A. Anagnostopoulos, L. Becchetti, I. Bordino, S. Leonardi, I. Mele, and P. Sankowski. Stochastic query covering for fast approximate document retrieval. *ACM Trans. Inf. Syst.*, 33(3):11:1–11:35, 2015. 177

[23] A. Andoni, A. Nikolov, K. Onak, and G. Yaroslavtsev. Parallel algorithms for geometric graph problems. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 574–583, 2014. 7, 18, 133, 135, 137, 138

[24] A. Andoni, C. Stein, Z. Song, Z. Wang, and P. Zhong. Parallel graph connectivity in log diameter rounds. *CoRR*, abs/1805.03055, 2018. 137, 138

[25] S. Assadi. Combinatorial auctions do need modest interaction. In *Proceedings of the 2017 ACM Conference on Economics and Computation, EC '17, Cambridge, MA,*

USA, June 26-30, 2017, pages 145–162, 2017. 23, 209, 212, 213, 215, 246, 249

[26] S. Assadi. Simple round compression for parallel vertex cover. *CoRR*, abs/1709.04599, 2017. 17, 119, 120, 134

[27] S. Assadi. Tight space-approximation tradeoff for the multi-pass streaming set cover problem. In *Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2017, Chicago, IL, USA, May 14-19, 2017*, pages 321–335, 2017. 20, 176, 194, 204, 208, 209

[28] S. Assadi, M. Bateni, A. Bernstein, V. S. Mirrokni, and C. Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. *CoRR*, abs/1711.03076, 2017. 121

[29] S. Assadi, M. Bateni, A. Bernstein, V. S. Mirrokni, and C. Stein. Coresets meet EDCS: algorithms for matching and vertex cover on massive graphs. In *Proceedings of the 30th Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019*, 2019. 16, 17, 51, 89, 98, 119, 124, 132, 134

[30] S. Assadi and S. Khanna. Randomized composable coresets for matching and vertex cover. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 3–12, 2017. 16, 87, 89, 93, 96, 108, 120, 134, 208

[31] S. Assadi and S. Khanna. Tight bounds on the round complexity of the distributed maximum coverage problem. In *Proceedings of the Twenty-Nine Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, 2018. 21, 207, 214, 216, 228, 243, 249

[32] S. Assadi, S. Khanna, and Y. Li. Tight bounds for single-pass streaming complexity of the set cover problem. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 698–711, 2016. 20, 176, 179, 186, 209

[33] S. Assadi, S. Khanna, and Y. Li. On estimating maximum matching size in graph streams. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1723–1742, 2017. 16, 51, 53, 55, 56, 57, 58, 86, 96, 209, 212, 215, 259

[34] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev. Tight bounds for linear sketches of approximate matchings. *CoRR*, abs/1505.01467, 2015. 55, 57, 67

[35] S. Assadi, S. Khanna, Y. Li, and G. Yaroslavtsev. Maximum matchings in dynamic graph streams and the simultaneous communication model. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1345–1364, 2016. 16, 51, 55, 57, 67, 90, 96, 208, 209, 212

[36] S. Assadi, X. Sun, and O. Weinstein. Massively parallel algorithms for finding well-connected components in sparse graphs. *CoRR*, abs/1805.02974, 2018. 18, 133, 138, 145

[37] J.-Y. Audibert and S. Bubeck. Best Arm Identification in Multi-Armed Bandits. In

COLT, 2010. 257

[38] G. Ausiello, N. Boria, A. Giannakos, G. Lucarelli, and V. T. Paschos. Online maximum k-coverage. *Discrete Applied Mathematics*, 160(13-14):1901–1913, 2012. 208, 209

[39] L. Babai, P. Frankl, and J. Simon. Complexity classes in communication complexity theory (preliminary version). In *27th Annual Symposium on Foundations of Computer Science, 27-29 October 1986*, pages 337–347, 1986. 41

[40] A. Badanidiyuru, B. Mirzasoleiman, A. Karbasi, and A. Krause. Streaming submodular maximization: massive data summarization on the fly. In *The 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '14, New York, NY, USA - August 24 - 27, 2014*, pages 671–680, 2014. 13, 18, 21, 90, 176, 177, 178, 207, 208, 209, 210, 211

[41] A. Badanidiyuru and J. Vondrák. Fast algorithms for maximizing submodular functions. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, January 5-7, 2014*, pages 1497–1514, 2014. 213

[42] B. Bahmani, R. Kumar, and S. Vassilvitskii. Densest subgraph in streaming and mapreduce. *PVLDB*, 5(5):454–465, 2012. 14

[43] M. Balcan, S. Ehrlich, and Y. Liang. Distributed k-means and k-median clustering on general communication topologies. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013.*, pages 1995–2003, 2013. 90, 93, 211

[44] Z. Bar-Yossef, T. S. Jayram, R. Kumar, and D. Sivakumar. An information statistics approach to data stream and communication complexity. In *43rd Symposium on Foundations of Computer Science (FOCS 2002), 16-19 November 2002, Proceedings*, pages 209–218, 2002. 41, 46, 49, 191, 192

[45] Z. Bar-Yossef, R. Kumar, and D. Sivakumar. Reductions in streaming algorithms, with an application to counting triangles in graphs. In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms, January 6-8, 2002, San Francisco, CA, USA.*, pages 623–632, 2002. 12

[46] B. Barak, M. Braverman, X. Chen, and A. Rao. How to compress interactive communication. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, 5-8 June 2010*, pages 67–76, 2010. 46, 49, 190, 195

[47] G. Barnes and U. Feige. Short random walks on graphs. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*, pages 728–737, 1993. 171

[48] S. Baswana. Streaming algorithm for graph spanners - single pass and constant processing time per edge. *Inf. Process. Lett.*, 106(3):110–114, 2008. 12

[49] M. Bateni, A. Bhaskara, S. Lattanzi, and V. S. Mirrokni. Distributed balanced clustering via mapping coresets. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December*

8-13 2014, pages 2591–2599, 2014. 90, 211

[50] M. Bateni, H. Esfandiari, and V. S. Mirrokni. Almost optimal streaming algorithms for coverage problems. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 13–23, 2017. 13, 18, 20, 176, 177, 178, 181, 201, 208, 209, 243

[51] M. Bateni, H. Esfandiari, and V. S. Mirrokni. Optimal distributed submodular optimization via sketching. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD 2018, August 19-23, 2018*, pages 1138–1147, 2018. 208

[52] R. Beals, H. Buhrman, R. Cleve, M. Mosca, and R. de Wolf. Quantum lower bounds by polynomials. *J. ACM*, 48(4), July 2001. 173

[53] P. Beame, P. Koutris, and D. Suciu. Communication steps for parallel query processing. In *Proceedings of the 32nd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2013, New York, NY, USA - June 22 - 27, 2013*, pages 273–284, 2013. 7, 15, 18, 96, 133, 135, 138, 211

[54] S. Behnezhad, M. Derakhshan, H. Esfandiari, E. Tan, and H. Yami. Brief announcement: Graph matching in massive datasets. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 133–136, 2017. 96

[55] S. Behnezhad, M. Derakhshan, and M. Hajiaghayi. Brief announcement: Semi-mapreduce meets congested clique. *CoRR*, abs/1802.10297, 2018. 14, 134, 137

[56] S. Behnezhad, M. Derakhshan, M. Hajiaghayi, and R. M. Karp. Massively parallel symmetry breaking on sparse graphs: MIS and maximal matching. *CoRR*, abs/1807.06701, 2018. 122, 138

[57] A. Bernstein and C. Stein. Fully dynamic matching in bipartite graphs. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, July 6-10, 2015, Proceedings, Part I*, pages 167–179, 2015. 96, 97, 98, 110, 121

[58] A. Bernstein and C. Stein. Faster fully dynamic matchings with small approximation ratios. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016*, pages 692–711, 2016. 97, 98, 110, 121

[59] Bertinoro workshop 2014, problem 64. http://sublinear.info/index.php?title=Open_Problems:64. Accessed: 2018-4-4. 15, 51, 54

[60] S. Bhattacharya, M. Henzinger, and G. F. Italiano. Deterministic fully dynamic data structures for vertex cover and matching. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, January 4-6, 2015*, pages 785–804, 2015. 94

[61] S. Bhattacharya, M. Henzinger, D. Nanongkai, and C. E. Tsourakakis. Space- and time-efficient algorithm for maintaining dense subgraphs on one-pass dynamic streams. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 173–182,

2015. 13, 89, 90

[62] Y. Birk, N. Linial, and R. Meshulam. On the uniform-traffic capacity of single-hop interconnections employing shared directional multichannels. *IEEE Transactions on Information Theory*, 39(1):186–191, 1993. 30

[63] G. E. Blelloch, R. Peng, and K. Tangwongsan. Linear-work greedy parallel approximate set cover and variants. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 23–32, 2011. 14, 208

[64] B. Bollobás. Random graphs. In *Modern graph theory*, pages 215–252. Springer, 1998. 102, 140

[65] S. Brandt, M. Fischer, and J. Uitto. Matching and MIS for uniformly sparse graphs in the low-memory MPC model. *CoRR*, abs/1807.05374, 2018. 122, 138

[66] M. Braverman. Interactive information complexity. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, May 19 - 22, 2012*, pages 505–524, 2012. 46, 190, 195

[67] M. Braverman, F. Ellen, R. Oshman, T. Pitassi, and V. Vaikuntanathan. A tight bound for set disjointness in the message-passing model. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 668–677, 2013. 13, 208

[68] M. Braverman, A. Garg, D. Pankratov, and O. Weinstein. From information to exact communication. In *Symposium on Theory of Computing Conference, STOC'13, June 1-4, 2013*, pages 151–160, 2013. 41, 191, 192

[69] M. Braverman, A. Garg, D. Pankratov, and O. Weinstein. Information lower bounds via self-reducibility. In *Computer Science - Theory and Applications - 8th International Computer Science Symposium in Russia, CSR 2013, June 25-29, 2013. Proceedings*, pages 183–194, 2013. 204

[70] M. Braverman, J. Mao, and S. M. Weinberg. Parallel Algorithms for Select and Partition with Noisy Comparisons. In *STOC*, 2016. 258

[71] M. Braverman, J. Mao, and S. M. Weinberg. On simultaneous two-player combinatorial auctions. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, January 7-10, 2018*, pages 2256–2273, 2018. 250

[72] M. Braverman and A. Moitra. An information complexity approach to extended formulations. In *Symposium on Theory of Computing Conference, STOC'13, June 1-4, 2013*, pages 161–170, 2013. 41

[73] M. Braverman and R. Oshman. A rounds vs. communication tradeoff for multiparty set disjointness. In *58th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2017, Berkeley, CA, USA, October 15-17, 2017*, pages 144–155, 2017. 14, 249

[74] M. Braverman and A. Rao. Information equals amortized communication. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, October 22-25, 2011*, pages 748–757, 2011. 47, 49, 190, 195

[75] M. Braverman, A. Rao, O. Weinstein, and A. Yehudayoff. Direct products in communication complexity. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013*, pages 746–755, 2013. 49

[76] M. Braverman and O. Weinstein. An interactive information odometer and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, June 14-17, 2015*, pages 341–350, 2015. 191, 198

[77] V. Braverman, R. Ostrovsky, and D. Vilenchik. How hard is counting triangles in the streaming model? In *Automata, Languages, and Programming - 40th International Colloquium, ICALP 2013, Riga, Latvia, July 8-12, 2013, Proceedings, Part I*, pages 244–254, 2013. 12

[78] P. Briest, P. Krysta, and B. Vöcking. Approximation techniques for utilitarian mechanism design. *SIAM J. Comput.*, 40(6):1587–1622, 2011. 249

[79] S. Bubeck, T. Wang, and N. Viswanathan. Multiple identifications in multi-armed bandits. In *ICML*, 2013. 257

[80] N. Buchbinder, M. Feldman, J. Naor, and R. Schwartz. A tight linear time (1/2)-approximation for unconstrained submodular maximization. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 649–658, 2012. 32

[81] L. Bulteau, V. Froese, K. Kutzkov, and R. Pagh. Triangle counting in dynamic graph streams. *Algorithmica*, 76(1):259–278, 2016. 90

[82] L. S. Buriol, G. Frahling, S. Leonardi, A. Marchetti-Spaccamela, and C. Sohler. Counting triangles in data streams. In *Proceedings of the Twenty-Fifth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 26-28, 2006, Chicago, Illinois, USA*, pages 253–262, 2006. 12

[83] M. Bury and C. Schwiegelshohn. Sublinear estimation of weighted matchings in dynamic data streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, September 14-16, 2015, Proceedings*, pages 263–274, 2015. 51, 52, 53, 56, 58, 59

[84] R. Busa-Fekete, B. Szorenyi, W. Cheng, P. Weng, and E. Hullermeier. Top-k selection based on adaptive sampling of noisy preferences. In *ICML*, 2013. 258

[85] G. Călinescu, C. Chekuri, M. Pál, and J. Vondrák. Maximizing a submodular set function subject to a matroid constraint (extended abstract). In *Integer Programming and Combinatorial Optimization, 12th International IPCO Conference, Ithaca, NY, USA, June 25-27, 2007, Proceedings*, pages 182–196, 2007. 32

[86] A. Chakrabarti, G. Cormode, and A. McGregor. Robust lower bounds for communication and stream computation. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008*, pages 641–650, 2008. 191

[87] A. Chakrabarti and S. Kale. Submodular maximization meets streaming: Matchings, matroids, and more. In *Integer Programming and Combinatorial Optimization - 17th International Conference, IPCO 2014, Bonn, Germany, June 23-25, 2014. Proceedings*, pages 210–221, 2014. 13, 208, 209

[88] A. Chakrabarti and O. Regev. An optimal lower bound on the communication com-

plexity of gap-hamming-distance. In *Proceedings of the 43rd ACM Symposium on Theory of Computing, STOC 2011, 6-8 June 2011*, pages 51–60, 2011. 41, 204

[89] A. Chakrabarti, Y. Shi, A. Wirth, and A. C. Yao. Informational complexity and the direct sum problem for simultaneous message complexity. In *42nd Annual Symposium on Foundations of Computer Science, FOCS 2001, 14-17 October 2001*, pages 270–278, 2001. 46, 49

[90] A. Chakrabarti and A. Wirth. Incidence geometries and the pass complexity of semi-streaming set cover. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1365–1373, 2016. 13, 18, 176, 177, 178, 179, 209, 213

[91] J. Chapman, B. Humphreys, M. Whiting, D. Miller, and R. Norris. Csiro askap science data archive: Overview, requirements and use cases. Technical report, ASKAP-SW-0017, 2014. 1

[92] J. Cheeger. A lower bound for the smallest eigenvalue of the laplacian. *Problems in analysis*, pages 195–199, 1970. 139

[93] C. Chekuri, S. Gupta, and K. Quanrud. Streaming algorithms for submodular function maximization. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, pages 318–330, 2015. 13

[94] J. Chen, H. L. Nguyen, and Q. Zhang. Submodular maximization over sliding windows. *CoRR*, abs/1611.00129, 2016. 208, 209, 210

[95] L. Chen and J. Li. On the Optimal Sample Complexity for Best Arm Identification. *arXiv preprint arXiv:1511.03774*, 2015. 257

[96] L. Chen, J. Li, and M. Qiao. Nearly Instance Optimal Sample Complexity Bounds for Top-k Arm Selection. *arXiv preprint arXiv:1702.03605*, 2017. 257

[97] Y. Chen and C. Suh. Spectral MLE: Top-k rank aggregation from pairwise comparisons. In *ICML*, 2015. 258, 259

[98] H. Chernoff. *Sequential analysis and optimal design*. SIAM, 1972. 260

[99] F. Chierichetti, R. Kumar, and A. Tomkins. Max-cover in map-reduce. In *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 231–240, 2010. 14, 20, 177, 208

[100] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan. One trillion edges: Graph processing at facebook-scale. *PVLDB*, 8(12):1804–1815, 2015. 1

[101] R. Chitnis, G. Cormode, H. Esfandiari, M. Hajiaghayi, A. McGregor, M. Monemizadeh, and S. Vorotnikova. Kernelization via sampling with applications to finding matchings and related problems in dynamic graph streams. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, January 10-12, 2016*, pages 1326–1344, 2016. 51, 52, 53, 56, 57, 90, 98

[102] R. H. Chitnis, G. Cormode, M. T. Hajiaghayi, and M. Monemizadeh. Parameterized streaming: Maximal matching and vertex cover. In *Proceedings of the Twenty-Sixth*

Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015, pages 1234–1251, 2015. 13, 51, 53, 89

[103] F. R. Chung. *Spectral graph theory*. Number 92. American Mathematical Soc., 1997. 139, 140

[104] K. L. Clarkson and D. P. Woodruff. Numerical linear algebra in the streaming model. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, May 31 - June 2, 2009*, pages 205–214, 2009. 56

[105] J. Cohen. Graph twiddling in a mapreduce world. *Computing in Science & Engineering*, 11(4):29–41, 2009. 137

[106] S. A. Cook, C. Dwork, and R. Reischuk. Upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 15(1):87–97, 1986. 133, 134, 174

[107] G. Cormode, H. Jowhari, M. Monemizadeh, and S. Muthukrishnan. The sparse awakens: Streaming algorithms for matching size estimation in sparse graphs. In *25th Annual European Symposium on Algorithms, ESA 2017, September 4-6, 2017*, pages 29:1–29:15, 2017. 57

[108] G. Cormode, H. J. Karloff, and A. Wirth. Set cover algorithms for very large datasets. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, pages 479–488, 2010. 13, 18, 176, 177, 209

[109] T. M. Cover and J. A. Thomas. *Elements of information theory (2. ed.)*. Wiley, 2006. 33, 34

[110] M. Crouch and D. S. Stubbs. Improved streaming algorithms for weighted matching, via unweighted matching. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014*, pages 96–104, 2014. 51, 54, 92

[111] A. Czumaj, J. Lacki, A. Madry, S. Mitrovic, K. Onak, and P. Sankowski. Round compression for parallel matching algorithms. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, June 25-29, 2018*, pages 471–484, 2018. 14, 17, 96, 119, 120, 121, 124, 134

[112] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward. The power of randomization: Distributed submodular maximization on massive datasets. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 1236–1244, 2015. 13, 14, 15, 208, 209, 210, 244

[113] R. da Ponte Barbosa, A. Ene, H. L. Nguyen, and J. Ward. A new framework for distributed submodular maximization. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, New Brunswick, New Jersey, USA*, pages 645–654, 2016. 14, 15, 21, 208, 211

[114] S. Davidson, S. Khanna, T. Milo, and S. Roy. Top-k and clustering with noisy comparisons. *ACM Transactions on Database Systems (TODS)*, 39(4):35, 2014. 258

[115] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters.

Commun. ACM, 51(1):107–113, 2008. 1, 6

[116] E. D. Demaine, P. Indyk, S. Mahabadi, and A. Vakilian. On streaming and communication complexity of the set cover problem. In *Distributed Computing - 28th International Symposium, DISC 2014, Austin, TX, USA, October 12-15, 2014. Proceedings*, pages 484–498, 2014. 13, 18, 19, 176, 177, 179, 181, 201, 202, 209

[117] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 624–633, 2014. 31

[118] S. Dobzinski. Computational efficiency requires simple taxation. In *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016*, pages 209–218, 2016. 96

[119] S. Dobzinski, N. Nisan, and S. Oren. Economic efficiency requires interaction. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 233–242, 2014. 22, 96, 209, 212, 246, 247, 248, 249

[120] S. Dobzinski, N. Nisan, and M. Schapira. Approximation algorithms for combinatorial auctions with complement-free bidders. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, May 22-24, 2005*, pages 610–618, 2005. 248, 249

[121] S. Dobzinski and M. Schapira. An improved approximation algorithm for combinatorial auctions with submodular bidders. In *Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2006, January 22-26, 2006*, pages 1064–1073, 2006. 249

[122] S. Dobzinski and J. Vondrák. From query complexity to computational complexity. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 1107–1116, 2012. 32

[123] D. P. Dubhashi and A. Panconesi. *Concentration of Measure for the Analysis of Randomized Algorithms*. Cambridge University Press, 2009. 25, 26

[124] P. Dütting and T. Kesselheim. Best-response dynamics in combinatorial auctions with item bidding. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, January 16-19*, pages 521–533, 2017. 249

[125] S. Eggert, L. Kliemann, and A. Srivastav. Bipartite graph matchings in the semi-streaming model. In *Algorithms - ESA 2009, 17th Annual European Symposium, September 7-9, 2009. Proceedings*, pages 492–503, 2009. 12, 51

[126] M. Elkin. Streaming and fully dynamic centralized algorithms for constructing and maintaining sparse spanners. *ACM Trans. Algorithms*, 7(2):20:1–20:17, 2011. 12

[127] M. Elkin and J. Zhang. Efficient algorithms for constructing (1+, varepsilon;, beta)-spanners in the distributed and streaming models. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*, pages 160–168, 2004. 12

[128] Y. Emek and A. Rosén. Semi-streaming set cover - (extended abstract). In *Automata, Languages, and Programming - 41st International Colloquium, ICALP 2014, Copen-*

hagen, Denmark, July 8-11, 2014, Proceedings, Part I, pages 453–464, 2014. 13, 18, 176, 177, 178, 179, 209

[129] A. Epasto, S. Lattanzi, S. Vassilvitskii, and M. Zadimoghaddam. Submodular optimization over sliding windows. In *Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Perth, Australia, April 3-7, 2017*, pages 421–430, 2017. 13, 18, 176, 208, 209, 210

[130] L. Epstein, A. Levin, J. Mestre, and D. Segev. Improved approximation guarantees for weighted matching in the semi-streaming model. *SIAM J. Discrete Math.*, 25(3):1251–1265, 2011. 51

[131] H. Esfandiari, M. T. Hajiaghayi, V. Liaghat, M. Monemizadeh, and K. Onak. Streaming algorithms for estimating the matching size in planar graphs and beyond. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, January 4-6, 2015*, pages 1217–1233, 2015. 51, 52, 53, 56, 59

[132] E. Even-Dar, S. Mannor, and Y. Mansour. Action elimination and stopping conditions for the multi-armed bandit and reinforcement learning problems. *Journal of Machine Learning Research*, 7:1079–1105, 2006. 257, 259

[133] M. Farach and M. Thorup. String matching in lempel—ziv compressed strings. *Algorithmica*, 20(4):388–404, Apr 1998. 135

[134] T. Feder, E. Kushilevitz, and M. Naor. Amortized communication complexity (preliminary version). In *32nd Annual Symposium on Foundations of Computer Science, 1-4 October 1991*, pages 239–248, 1991. 49

[135] U. Feige. A threshold of ln n for approximating set cover. *J. ACM*, 45(4):634–652, 1998. 31, 33

[136] U. Feige. On maximizing welfare when utility functions are subadditive. *SIAM J. Comput.*, 39(1):122–142, 2009. 249

[137] U. Feige, P. Raghavan, D. Peleg, and E. Upfal. Computing with Noisy Information. *SIAM Journal on Computing*, 23(5):1001–1018, 1994. 258

[138] U. Feige and J. Vondrák. Approximation algorithms for allocation problems: Improving the factor of 1 - 1/e. In *47th Annual IEEE Symposium on Foundations of Computer Science (FOCS 2006), 21-24 October 2006, Proceedings*, pages 667–676, 2006. 249

[139] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. 12, 51, 53, 92, 95

[140] J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the data-stream model. *SIAM J. Comput.*, 38(5):1709–1727, 2008. 12

[141] J. Feigenbaum, S. Kannan, M. Strauss, and M. Viswanathan. An approximate l^1-difference algorithm for massive data streams. In *40th Annual Symposium on Foundations of Computer Science, FOCS '99, 17-18 October, 1999, New York, NY, USA*, pages 501–511, 1999. 5

[142] M. Feldman, J. Naor, and R. Schwartz. A unified continuous greedy algorithm for submodular maximization. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 570–579, 2011. 32

[143] E. Fischer, E. Lehman, I. Newman, S. Raskhodnikova, R. Rubinfeld, and A. Samorodnitsky. Monotonicity testing over general poset domains. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing, May 19-21, 2002, Montréal, Québec, Canada*, pages 474–483, 2002. 30, 56

[144] M. Fischer and J. Uitto. Breaking the linear-memory barrier in mpc: Fast mis on trees with n^{ε} memory per machine. *arXiv preprint arXiv:1802.06748*, 2018. 137

[145] P. Flajolet and G. N. Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, 1985. 5

[146] J. Fox. A new proof of the graph removal lemma. *Annals of Mathematics*, 174(1):561–579, 2011. 30

[147] J. Fox, H. Huang, and B. Sudakov. On graphs decomposable into induced matchings of linear sizes. *arXiv preprint arXiv:1512.07852*, 2015. 30, 95

[148] G. Frahling, P. Indyk, and C. Sohler. Sampling in dynamic data streams and applications. *International Journal of Computational Geometry & Applications*, 18(01n02):3–28, 2008. 57

[149] J. Friedman. A proof of alon's second eigenvalue conjecture. In *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, pages 720–724. ACM, 2003. 146

[150] V. Gabillon, M. Ghavamzadeh, and A. Lazaric. Best arm identification: A unified approach to fixed budget and fixed confidence. In *NIPS*, 2012. 257

[151] A. Ganor, G. Kol, and R. Raz. Exponential separation of information and communication for boolean functions. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, June 14-17, 2015*, pages 557–566, 2015. 49, 58

[152] A. Ganor, G. Kol, and R. Raz. Exponential separation of communication and external information. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, June 18-21, 2016*, pages 977–986, 2016. 58

[153] D. Gavinsky, J. Kempe, I. Kerenidis, R. Raz, and R. de Wolf. Exponential separations for one-way quantum communication complexity, with applications to cryptography. *STOC*, pages 516–525, 2007. 40, 58

[154] H. Gazit. An optimal randomized parallel algorithm for finding connected components in a graph. In *Foundations of Computer Science, 1986., 27th Annual Symposium on*, pages 492–501. IEEE, 1986. 133, 134, 135, 137

[155] M. Ghaffari, T. Gouleakis, C. Konrad, S. Mitrovic, and R. Rubinfeld. Improved massively parallel computation algorithms for mis, matching, and vertex cover. In *Proceedings of the 2018 ACM Symposium on Principles of Distributed Computing, PODC 2018, July 23-27, 2018*, pages 129–138, 2018. 121

[156] M. Ghaffari and M. Parter. MST in log-star rounds of congested clique. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 19–28, 2016. 14, 134, 137

[157] M. Ghaffari and J. Uitto. Sparsifying distributed algorithms with ramifications in massively parallel computation and centralized local computation. *CoRR*, abs/1807.06251, 2018. 121, 138

[158] A. L. Gibbs and F. E. Su. On choosing and bounding probability metrics. *International statistical review*, 70(3):419–435, 2002. 259

[159] C. Gkantsidis, M. Mihail, and A. Saberi. Conductance and congestion in power law graphs. In *Proceedings of the International Conference on Measurements and Modeling of Computer Systems, SIGMETRICS 2003, June 9-14, 2003, San Diego, CA, USA*, pages 148–159, 2003. 18, 135

[160] A. Goel, M. Kapralov, and S. Khanna. On the communication and streaming complexity of maximum bipartite matching. In *Proceedings of the Twenty-third Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '12, pages 468–485. SIAM, 2012. 12, 30, 51, 53, 54, 95

[161] M. T. Goodrich, N. Sitchinava, and Q. Zhang. Sorting, searching, and simulation in the mapreduce framework. In *Algorithms and Computation - 22nd International Symposium, ISAAC 2011, Yokohama, Japan, December 5-8, 2011. Proceedings*, pages 374–383, 2011. 7

[162] M. Göös, T. S. Jayram, T. Pitassi, and T. Watson. Randomized communication vs. partition number. In *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017*, pages 52:1–52:15, 2017. 191, 198

[163] W. Gowers. Some unsolved problems in additive/combinatorial number theory. *preprint*, 2001. 30, 95

[164] T. Grossman and A. Wool. Computational experience with approximation algorithms for the set covering problem. *European Journal of Operational Research*, 101(1):81–92, 1997. 177

[165] D. V. Gucht, R. Williams, D. P. Woodruff, and Q. Zhang. The communication complexity of distributed set-joins with applications to matrix multiplication. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 199–212, 2015. 13, 208

[166] S. Guha, Y. Li, and Q. Zhang. Distributed partial clustering. In *Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2017, Washington DC, USA, July 24-26, 2017*, pages 143–152, 2017. 13, 208

[167] S. Guha and A. McGregor. Tight lower bounds for multi-pass stream computation via pass elimination. In *Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games*, pages 760–772, 2008. 43

[168] S. Guha and A. McGregor. Stream order and order statistics: Quantile estimation in

random-order streams. *SIAM J. Comput.*, 38(5):2044–2059, 2009. 181

[169] S. Guha, A. McGregor, and D. Tench. Vertex and hyperedge connectivity in dynamic graph streams. In *Proceedings of the 34th ACM Symposium on Principles of Database Systems, PODS 2015, Melbourne, Victoria, Australia, May 31 - June 4, 2015*, pages 241–247, 2015. 13, 89

[170] V. Guruswami and K. Onak. Superlinear lower bounds for multipass graph processing. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 287–298, 2013. 12, 14, 51

[171] B. V. Halldórsson, M. M. Halldórsson, E. Losievskaja, and M. Szegedy. Streaming algorithms for independent sets. In *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part I*, pages 641–652, 2010. 12

[172] M. M. Halldórsson, X. Sun, M. Szegedy, and C. Wang. Streaming and communication complexity of clique approximation. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part I*, pages 449–460, 2012. 12

[173] S. Halperin and U. Zwick. An optimal randomized logarithmic time connectivity algorithm for the EREW PRAM (extended abstract). In *SPAA*, pages 1–10, 1994. 133, 134, 135, 137, 142

[174] S. Har-Peled, P. Indyk, S. Mahabadi, and A. Vakilian. Towards tight bounds for the streaming set cover problem. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 371–383, 2016. 13, 18, 19, 176, 177, 178, 179, 180, 181, 191, 201, 202, 209

[175] N. J. A. Harvey, C. Liaw, and P. Liu. Greedy and local ratio algorithms in the mapreduce model. In *Proceedings of the 30th on Symposium on Parallelism in Algorithms and Architectures, SPAA 2018, July 16-18, 2018*, pages 43–52, 2018. 96

[176] J. Håstad and A. Wigderson. Simple analysis of graph tests for linearity and PCP. *Random Struct. Algorithms*, 22(2):139–160, 2003. 30

[177] R. Heckel, N. B. Shah, K. Ramchandran, and M. J. Wainwright. Active Ranking from Pairwise Comparisons and when Parametric Assumptions Dont Help. *arXiv preprint arXiv:1606.08842*, 2016. 258

[178] J. W. Hegeman, G. Pandurangan, S. V. Pemmaraju, V. B. Sardeshmukh, and M. Scquizzato. Toward optimal bounds in the congested clique: Graph connectivity and MST. In *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 91–100, 2015. 14, 134, 137

[179] M. R. Henzinger, P. Raghavan, and S. Rajagopalan. Computing on data streams. In *External Memory Algorithms, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, May 20-22, 1998*, pages 107–118, 1998. 5, 12

[180] S. Hoory, N. Linial, and A. Wigderson. Expander graphs and their applications.

Bulletin of the American Mathematical Society, 43(4):439–561, 2006. 142, 145

[181] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM J. Comput.*, 2(4):225–231, 1973. 28

[182] Z. Huang and P. Peng. Dynamic graph stream algorithms in o(n) space. In *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, pages 18:1–18:16, 2016. 13, 89

[183] Z. Huang, B. Radunovic, M. Vojnovic, and Q. Zhang. Communication complexity of approximate matching in distributed graphs. In *32nd International Symposium on Theoretical Aspects of Computer Science, STACS 2015, March 4-7, 2015, Garching, Germany*, pages 460–473, 2015. 13

[184] S. Im and B. Moseley. Brief announcement: Fast and better distributed mapreduce algorithms for k-center clustering. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, June 13-15, 2015*, pages 65–67, 2015. 211

[185] R. Impagliazzo and V. Kabanets. Constructive proofs of concentration bounds. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 13th International Workshop, APPROX 2010, and 14th International Workshop, RANDOM 2010, September 1-3, 2010. Proceedings*, pages 617–631, 2010. 181

[186] P. Indyk, S. Mahabadi, M. Mahdian, and V. S. Mirrokni. Composable core-sets for diversity and coverage maximization. In *Proceedings of the 33rd ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS'14, Snowbird, UT, USA, June 22-27, 2014*, pages 100–108, 2014. 9, 13, 14, 15, 20, 90, 93, 208, 209, 211

[187] P. Indyk, S. Mahabadi, and A. Vakilian. Towards tight bounds for the streaming set cover problem. *CoRR*, abs/1509.00118, 2015. 13, 18, 19, 176, 177, 179

[188] P. Indyk and D. P. Woodruff. Tight lower bounds for the distinct elements problem. In *44th Symposium on Foundations of Computer Science (FOCS 2003), 11-14 October 2003, Proceedings*, pages 283–288, 2003. 41, 204

[189] R. Jacob, T. Lieber, and N. Sitchinava. On the complexity of list ranking in the parallel external memory model. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 384–395, 2014. 15, 211

[190] R. Jain, A. Pereszlényi, and P. Yao. A direct product theorem for the two-party bounded-round public-coin communication complexity. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, October 20-23, 2012*, pages 167–176, 2012. 49

[191] R. Jain, J. Radhakrishnan, and P. Sen. A direct sum theorem in communication complexity via message compression. In *Automata, Languages and Programming, 30th International Colloquium, ICALP 2003, June 30 - July 4, 2003. Proceedings*, pages 300–315, 2003. 58

[192] K. Jamieson, M. Malloy, R. Nowak, and S. Bubeck. On Finding the Largest Mean

Among Many. *arXiv preprint arXiv:1306.3917v1*, 2013. 257

[193] M. Jang, S. Kim, C. Suh, and S. Oh. Top-k Ranking from Pairwise Comparisons: When Spectral Ranking is Optimal. *arXiv preprint arXiv:1603.04153*, 2016. 258

[194] D. S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974. 31

[195] H. Jowhari, M. Sağlam, and G. Tardos. 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*, pages 49–58. ACM, 2011. 57

[196] K.-S. Jun, K. Jamieson, R. Nowak, and X. Zhu. Top Arm Identification in Multi-Armed Bandits with Batch Arm Pulls. In *AISTATS*, 2016. 257

[197] T. Jurdzinski and K. Nowicki. MST in $O(1)$ rounds of congested clique. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018, New Orleans, LA, USA, January 7-10, 2018*, pages 2620–2632, 2018. 14, 134, 137

[198] S. Kale and S. Tirodkar. Maximum matching in two, three, and a few more passes over graph streams. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2017, August 16-18, 2017, Berkeley, CA, USA*, pages 15:1–15:21, 2017. 12

[199] S. Kalyanakrishnan and P. Stone. Efficient Selection of Multiple Bandit Arms: Theory and Practice. In *ICML*, 2010. 257, 259

[200] S. Kalyanakrishnan, A. Tewari, P. Auer, and P. Stone. PAC Subset Selection in Stochastic Multi-armed Bandits. In *ICML*, 2012. 257

[201] B. Kalyanasundaram and G. Schnitger. The probabilistic communication complexity of set intersection. *SIAM J. Discrete Math.*, 5(4):545–557, 1992. 41

[202] D. M. Kane, K. Mehlhorn, T. Sauerwald, and H. Sun. Counting arbitrary subgraphs in data streams. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Warwick, UK, July 9-13, 2012, Proceedings, Part II*, pages 598–609, 2012. 12

[203] U. Kang, C. E. Tsourakakis, and C. Faloutsos. Pegasus: A peta-scale graph mining system implementation and observations. In *Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on*, pages 229–238. IEEE, 2009. 137

[204] M. Kapralov. Better bounds for matchings in the streaming model. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1679–1697, 2013. 12, 51, 53, 54, 95

[205] M. Kapralov, S. Khanna, and M. Sudan. Approximating matching size from random streams. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 734–751, 2014. 16, 51, 52, 53, 55, 94, 181

[206] M. Kapralov, S. Khanna, and M. Sudan. Streaming lower bounds for approximating MAX-CUT. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 1263–1282, 2015. 12

[207] M. Kapralov, S. Khanna, M. Sudan, and A. Velingker. $(1 + \Omega(1))$-approximation to MAX-CUT requires linear space. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1703–1722, 2017. 12

[208] M. Kapralov, Y. T. Lee, C. Musco, C. Musco, and A. Sidford. Single pass spectral sparsification in dynamic streams. In *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*, pages 561–570, 2014. 13, 89, 90

[209] M. Kapralov and D. P. Woodruff. Spanners and sparsifiers in dynamic streams. In *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 272–281, 2014. 11, 13, 89, 90, 208, 211

[210] M. Karchmer, E. Kushilevitz, and N. Nisan. Fractional covers and communication complexity. *SIAM J. Discrete Math.*, 8(1):76–92, 1995. 49

[211] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via the direct sum in communication complexity. *Computational Complexity*, 5(3/4):191–204, 1995. 49

[212] D. R. Karger, N. Nisan, and M. Parnas. Fast connected components algorithms for the EREW PRAM. In *SPAA*, pages 373–381, 1992. 133, 134, 135, 137, 142

[213] H. J. Karloff, S. Suri, and S. Vassilvitskii. A model of computation for mapreduce. In *Proceedings of the Twenty-First Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2010, Austin, Texas, USA, January 17-19, 2010*, pages 938–948, 2010. 7, 14, 18, 134, 135, 137, 138, 143

[214] Z. Karnin, T. Koren, and O. Somekh. Almost Optimal Exploration in Multi-Armed Bandits. In *ICML*, 2013. 257

[215] R. M. Karp. Reducibility among combinatorial problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York.*, pages 85–103, 1972. 31

[216] E. Kaufmann, O. Cappé, and A. Garivier. On the Complexity of Best-Arm Identification in Multi-Armed Bandit Models. *Journal of Machine Learning Research*, 17:1–42, 2016. 257

[217] J. A. Kelner and A. Levin. Spectral sparsification in the semi-streaming setting. In *28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, March 10-12, 2011, Dortmund, Germany*, pages 440–451, 2011. 12

[218] R. Kiveris, S. Lattanzi, V. S. Mirrokni, V. Rastogi, and S. Vassilvitskii. Connected components in mapreduce and beyond. In *Proceedings of the ACM Symposium on Cloud Computing, Seattle, WA, USA, November 03 - 05, 2014*, pages 18:1–18:13,

2014. 18, 135, 137, 143

[219] H. Klauck. A strong direct product theorem for disjointness. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, 5-8 June 2010*, pages 77–86, 2010. 49

[220] C. Konrad. Maximum matching in turnstile streams. In *Algorithms - ESA 2015 - 23rd Annual European Symposium, September 14-16, 2015, Proceedings*, pages 840–852, 2015. 51, 57, 96, 209, 212

[221] C. Konrad, F. Magniez, and C. Mathieu. Maximum matching in semi-streaming with few passes. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 231–242, 2012. 12, 51, 95, 181

[222] I. Kremer, N. Nisan, and D. Ron. On randomized one-round communication complexity. In *Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 29 May-1 June 1995, Las Vegas, Nevada, USA*, pages 596–605, 1995. 40

[223] R. Kumar, B. Moseley, S. Vassilvitskii, and A. Vattani. Fast greedy algorithms in mapreduce and streaming. In *25th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '13, Montreal, QC, Canada - July 23 - 25, 2013*, pages 1–10, 2013. 10, 11, 13, 14, 15, 20, 21, 134, 207, 208, 210, 211, 213, 214

[224] E. Kushilevitz and N. Nisan. *Communication complexity*. Cambridge University Press, 1997. 36, 49

[225] S. Lattanzi, B. Moseley, S. Suri, and S. Vassilvitskii. Filtering: a method for solving graph problems in mapreduce. In *SPAA 2011: Proceedings of the 23rd Annual ACM Symposium on Parallelism in Algorithms and Architectures, San Jose, CA, USA, June 4-6, 2011 (Co-located with FCRC 2011)*, pages 85–94, 2011. 10, 14, 96, 119, 120, 134, 137, 211

[226] R. Lavi and C. Swamy. Truthful and near-optimal mechanism design via linear programming. *J. ACM*, 58(6):25:1–25:24, 2011. 249

[227] B. Lehmann, D. J. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006. 249

[228] C. Lenzen. Optimal deterministic routing and sorting on the congested clique. In *ACM Symposium on Principles of Distributed Computing, PODC '13, Montreal, QC, Canada, July 22-24, 2013*, pages 42–50, 2013. 14, 134

[229] Y. Li, H. L. Nguyen, and D. P. Woodruff. On sketching matrix norms and the top singular vector. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, January 5-7, 2014*, pages 1562–1581, 2014. 56

[230] Y. Li, H. L. Nguyen, and D. P. Woodruff. Turnstile streaming algorithms might as well be linear sketches. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 174–183, 2014. 11, 44, 45, 54, 65, 70, 77

[231] Y. Li, X. Sun, C. Wang, and D. P. Woodruff. On the communication complexity of

linear algebraic problems in the message passing model. In *Distributed Computing - 28th International Symposium, DISC 2014, October 12-15, 2014. Proceedings*, pages 499–513, 2014. 56

[232] Y. Li and D. P. Woodruff. On approximating functions of the singular values in a stream. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 726–739, 2016. 16, 56, 57, 58

[233] Y. Li and D. P. Woodruff. Tight bounds for sketching the operator norm, schatten norms, and subspace embeddings. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016*, pages 39:1–39:11, 2016. 56

[234] Z. Lotker, B. Patt-Shamir, and S. Pettie. Improved distributed approximate matching. *J. ACM*, 62(5):38:1–38:17, 2015. 121

[235] Z. Lotker, E. Pavlov, B. Patt-Shamir, and D. Peleg. MST construction in o(log log n) communication rounds. In *SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, June 7-9, 2003, San Diego, California, USA (part of FCRC 2003)*, pages 94–100, 2003. 14, 134, 137

[236] L. Lovász. On the ratio of optimal integral and fractional covers. *Discrete mathematics*, 13(4):383–390, 1975. 249

[237] L. Lovász and M. D. Plummer. *Matching theory*, volume 367. American Mathematical Soc., 2009. 27, 56

[238] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. ACM*, 41(5):960–981, 1994. 31

[239] F. D. Malliaros and V. Megalooikonomou. Expansion properties of large social graphs. In *Database Systems for Adanced Applications - 16th International Conference, DASFAA 2011, International Workshops: GDB, SIM3, FlashDB, SNSMW, DaMEN, DQIS, Hong Kong, China, April 22-25, 2011. Proceedings*, pages 311–322, 2011. 18, 135

[240] S. Mannor and J. N. Tsitsiklis. The Sample Complexity of Exploration in the Multi-Armed Bandit Problem. *Journal of Machine Learning Research*, 5:623–648, 2004. 257

[241] A. McGregor. Finding graph matchings in data streams. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, Berkeley, CA, USA, August 22-24, 2005, Proceedings*, pages 170–181, 2005. 12, 51, 132

[242] A. McGregor. Graph stream algorithms: a survey. *SIGMOD Record*, 43(1):9–20, 2014. 2, 9, 51, 53, 56

[243] A. McGregor, D. Tench, S. Vorotnikova, and H. T. Vu. Densest subgraph in dynamic graph streams. In *Mathematical Foundations of Computer Science 2015 - 40th In-*

ternational Symposium, MFCS 2015, Milan, Italy, August 24-28, 2015, Proceedings, Part II, pages 472–482, 2015. 13, 89, 90

[244] A. McGregor and S. Vorotnikova. Planar matching in streams revisited. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2016, September 7-9, 2016*, pages 17:1–17:12, 2016. 51, 52, 53, 56

[245] A. McGregor and S. Vorotnikova. A simple, space-efficient, streaming algorithm for matchings in low arboricity graphs. In *1st Symposium on Simplicity in Algorithms, SOSA 2018, January 7-10, 2018*, pages 14:1–14:4, 2018. 57

[246] A. McGregor, S. Vorotnikova, and H. T. Vu. Better algorithms for counting triangles in data streams. In *Proceedings of the 35th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, PODS 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 401–411, 2016. 12

[247] A. McGregor and H. T. Vu. Better streaming algorithms for the maximum coverage problem. In *20th International Conference on Database Theory, ICDT 2017, March 21-24, 2017, Venice, Italy*, pages 22:1–22:18, 2017. 13, 18, 20, 21, 176, 177, 178, 181, 201, 202, 207, 208, 209, 210, 213, 243

[248] S. Micali and V. V. Vazirani. An o(sqrt(|v|) |e|) algorithm for finding maximum matching in general graphs. In *21st Annual Symposium on Foundations of Computer Science, Syracuse, New York, USA, 13-15 October 1980*, pages 17–27, 1980. 28

[249] V. S. Mirrokni and M. Zadimoghaddam. Randomized composable core-sets for distributed submodular maximization. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 153–162, 2015. 13, 14, 15, 20, 90, 93, 96, 208, 244

[250] B. Mirzasoleiman, A. Karbasi, R. Sarkar, and A. Krause. Distributed submodular maximization: Identifying representative elements in massive data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013, Lake Tahoe, Nevada, United States.*, pages 2049–2057, 2013. 14, 90, 208

[251] R. H. Morris. Counting large numbers of events in small registers. *Commun. ACM*, 21(10):840–842, 1978. 5

[252] D. Moshkovitz. The projection games conjecture and the np-hardness of ln n-approximating set-cover. *Theory of Computing*, 11:221–235, 2015. 31

[253] J. I. Munro and M. Paterson. Selection and sorting with limited storage. In *19th Annual Symposium on Foundations of Computer Science, Ann Arbor, Michigan, USA, 16-18 October 1978*, pages 253–258, 1978. 5

[254] S. Muthukrishnan. *Data streams: Algorithms and applications*. Now Publishers Inc, 2005. 5, 12, 92, 177

[255] D. Nanongkai, A. D. Sarma, and G. Pandurangan. A tight unconditional lower bound on distributed randomwalk computation. In *Proceedings of the 30th Annual ACM Symposium on Principles of Distributed Computing, PODC 2011, June 6-8, 2011*,

pages 257–266, 2011. 142

[256] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher. An analysis of approximations for maximizing submodular set functions - I. *Math. Program.*, 14(1):265–294, 1978. 31, 32

[257] I. Newman. Private vs. common random bits in communication complexity. *Inf. Process. Lett.*, 39(2):67–71, 1991. 38

[258] N. Nisan. The communication complexity of approximate set packing and covering. In *Automata, Languages and Programming, 29th International Colloquium, ICALP 2002, July 8-13, 2002, Proceedings*, pages 868–875, 2002. 180, 181, 249

[259] N. Nisan and I. Segal. The communication requirements of efficient allocations and supporting prices. *J. Economic Theory*, 129(1):192–224, 2006. 249

[260] N. Nisan and M. Szegedy. On the degree of boolean functions as real polynomials. *Comput. Complex.*, 4(4), Oct. 1994. 173

[261] A. Norouzi-Fard, J. Tarnawski, S. Mitrovic, A. Zandieh, A. Mousavifar, and O. Svensson. Beyond 1/2-approximation for submodular maximization on massive data streams. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, July 10-15, 2018*, pages 3826–3835, 2018. 208

[262] K. Onak. Round compression for parallel graph algorithms in strongly sublinear space. *CoRR*, abs/1807.08745, 2018. 121, 138

[263] K. Onak and R. Rubinfeld. Maintaining a large matching and a small vertex cover. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, 5-8 June 2010*, pages 457–464, 2010. 94, 131

[264] A. Panconesi and A. Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997. 181, 182

[265] I. Parberry and P. Y. Yan. Improved upper and lower time bounds for parallel random access machines without simultaneous writes. *SIAM J. Comput.*, 20(1):88–99, 1991. 133, 134, 174

[266] M. Parnas and D. Ron. Approximating the minimum vertex cover in sublinear time and a connection to distributed algorithms. *Theor. Comput. Sci.*, 381(1-3):183–196, 2007. 94, 103, 104, 120, 131

[267] A. Paz and G. Schwartzman. A $(2 + \varepsilon)$-approximation for maximum weight matching in the semi-streaming model. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 2153–2161, 2017. 12, 51

[268] J. M. Phillips, E. Verbin, and Q. Zhang. Lower bounds for number-in-hand multiparty communication complexity, made easy. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 486–501, 2012. 6, 13, 208

[269] A. Pietracaprina, G. Pucci, M. Riondato, F. Silvestri, and E. Upfal. Space-round

tradeoffs for mapreduce computations. In *International Conference on Supercomputing, ICS'12, Italy*, pages 235–244, 2012. 15, 211

[270] A. Rao and M. Sinha. Simplified separation of information and communication. *Electronic Colloquium on Computational Complexity (ECCC)*, 22:57, 2015. 49, 58

[271] V. Rastogi, A. Machanavajjhala, L. Chitnis, and A. D. Sarma. Finding connected components in map-reduce in logarithmic rounds. In *29th IEEE International Conference on Data Engineering, ICDE 2013, April 8-12, 2013*, pages 50–61, 2013. 18, 133, 135, 137, 143

[272] A. A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992. 41

[273] J. Reif. Optimal parallel algorithms for interger sorting and graph connectivity. technical report. Technical report, Harvard Univ., Cambridge, MA (USA). Aiken Computation Lab., 1985. 133, 134, 135, 137

[274] O. Reingold. Undirected st-connectivity in log-space. In *Proceedings of the 37th Annual ACM Symposium on Theory of Computing, May 22-24, 2005*, pages 376–385, 2005. 141, 142

[275] O. Reingold, L. Trevisan, and S. P. Vadhan. Pseudorandom walks on regular digraphs and the RL vs. L problem. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, May 21-23, 2006*, pages 457–466, 2006. 142, 145

[276] O. Reingold, S. P. Vadhan, and A. Wigderson. Entropy waves, the zig-zag graph product, and new constant-degree expanders and extractors. In *41st Annual Symposium on Foundations of Computer Science, FOCS 2000, 12-14 November 2000, Redondo Beach, California, USA*, pages 3–13, 2000. 137, 141, 142, 145

[277] T. Roughgarden, S. Vassilvitskii, and J. R. Wang. Shuffles and circuits: (on lower bounds for modern parallel computation). In *Proceedings of the 28th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA 2016, Asilomar State Beach/Pacific Grove, CA, USA, July 11-13, 2016*, pages 1–12, 2016. 7, 15, 18, 133, 135, 136, 172, 173, 211

[278] E. Rozenman and S. P. Vadhan. Derandomized squaring of graphs. In *Approximation, Randomization and Combinatorial Optimization, Algorithms and Techniques, 8th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2005 and 9th InternationalWorkshop on Randomization and Computation, RANDOM 2005, August 22-24, 2005, Proceedings*, pages 436–447, 2005. 142, 145

[279] I. Z. Ruzsa and E. Szemerédi. Triple systems with no six points carrying three triangles. *Combinatorics (Keszthely, 1976), Coll. Math. Soc. J. Bolyai*, 18:939–945, 1978. 29, 30, 54, 212

[280] M. Saglam. *Tight bounds for data stream algorithms and communication problems.* PhD thesis, Simon Fraser University, 2011. 186

[281] B. Saha and L. Getoor. On maximum coverage in the streaming model & application to multi-topic blog-watch. In *Proceedings of the SIAM International Conference on*

Data Mining, SDM 2009, Sparks, Nevada, USA, pages 697–708, 2009. 13, 18, 176, 177, 201, 208, 209

[282] A. D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. In *Proceedings of the Twenty-Seventh ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2008, June 9-11, 2008, Vancouver, BC, Canada*, pages 69–78, 2008. 12

[283] A. D. Sarma, S. Gollapudi, and R. Panigrahy. Estimating pagerank on graph streams. *J. ACM*, 58(3):13:1–13:19, 2011. 137, 142

[284] A. D. Sarma, D. Nanongkai, G. Pandurangan, and P. Tetali. Distributed random walks. *J. ACM*, 60(1):2:1–2:31, 2013. 137, 142

[285] J. P. Schmidt, A. Siegel, and A. Srinivasan. Chernoff-hoeffding bounds for applications with limited independence. *SIAM J. Discrete Math.*, 8(2):223–250, 1995. 26

[286] A. Schrijver. *Combinatorial optimization: polyhedra and efficiency*, volume 24. Springer Science & Business Media, 2003. 27

[287] N. B. Shah and M. J. Wainwright. Simple, Robust and Optimal Ranking from Pairwise Comparisons. *arXiv preprint arXiv:1512.08949*, 2015. 258, 259

[288] Y. Shiloach and U. Vishkin. An o (logn) parallel connectivity algorithm. *Journal of Algorithms*, 3(1):57–67, 1982. 133, 134, 135, 137

[289] P. Slavík. A tight analysis of the greedy algorithm for set cover. *J. Algorithms*, 25(2):237–254, 1997. 31

[290] T. Tao and V. H. Vu. *Additive combinatorics*, volume 105. Cambridge University Press, 2006. 30

[291] L. Trevisan. Lecture notes on expansion, sparsest cut, and spectral graph theory, 2013. 142, 145

[292] W. T. Tutte. The factorization of linear graphs. *Journal of the London Mathematical Society*, 1(2):107–111, 1947. 56

[293] L. G. Valiant. A theory of the learnable. In *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, pages 436–445, 1984. 256

[294] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990. 7

[295] E. Verbin and W. Yu. The streaming complexity of cycle counting, sorting by reversals, and other problems. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2011, January 23-25, 2011*, pages 11–25, 2011. 40, 58

[296] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the 40th Annual ACM Symposium on Theory of Computing, May 17-20, 2008*, pages 67–74, 2008. 249

[297] O. Weinstein. Information complexity and the quest for interactive compression.

SIGACT News, 46(2):41–64, 2015. 46, 47, 49

[298] O. Weinstein and D. P. Woodruff. The simultaneous communication of disjointness with applications to data streams. In *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, July 6-10, 2015, Proceedings, Part I*, pages 1082–1093, 2015. 191

[299] T. White. *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 1st edition, 2009. 6

[300] D. P. Woodruff. Personal Communication. 2018-04-15. 45

[301] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014. 9

[302] D. P. Woodruff and Q. Zhang. Tight bounds for distributed functional monitoring. In *Proceedings of the 44th Symposium on Theory of Computing Conference, STOC 2012, New York, NY, USA, May 19 - 22, 2012*, pages 941–960, 2012. 13, 208

[303] D. P. Woodruff and Q. Zhang. When distributed computation is communication expensive. In *Distributed Computing - 27th International Symposium, DISC 2013, Jerusalem, Israel, October 14-18, 2013. Proceedings*, pages 16–30, 2013. 13, 208, 213

[304] D. P. Woodruff and Q. Zhang. An optimal lower bound for distinct elements in the message passing model. In *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA*, pages 718–733, 2014. 13, 208

[305] A. C. Yao. Some complexity questions related to distributive computing (preliminary report). In *Proceedings of the 11h Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1979, Atlanta, Georgia, USA*, pages 209–213, 1979. 36

[306] A. C. Yao. Lower bounds by probabilistic arguments (extended abstract). In *24th Annual Symposium on Foundations of Computer Science, Tucson, Arizona, USA, 7-9 November 1983*, pages 420–428, 1983. 38

[307] G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p-distances. *arXiv preprint arXiv:1710.01431*, 2017. 18

[308] G. Yaroslavtsev and A. Vadapalli. Massively parallel algorithms and hardness for single-linkage clustering under ℓ_p-distances. *arXiv preprint arXiv:1710.01431*, 2017. 135, 138

[309] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association. 6

[310] M. Zelke. Intractability of min- and max-cut in streaming graphs. *Inf. Process. Lett.*, 111(3):145–150, 2011. 12

[311] M. Zelke. Weighted matching in the semi-streaming model. *Algorithmica*, 62(1-2):1–20, 2012. 51