COMPUTING WITH DISTRIBUTED INFORMATION

Yang Li

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

2017

Supervisor of Dissertation

Sanjeev Khanna, Henry Salvatori Professor
Computer and Information Science

Co-Supervisor of Dissertation

Boon Thau Loo, Associate Professor
Computer and Information Science

Co-Supervisor of Dissertation

Linh T. X. Phan, Assistant Professor
Computer and Information Science

Graduate Group Chairperson

Lyle Ungar, Professor
Computer and Information Science

Dissertation Committee
Aaron Roth (Chair), Class of 1940 Bicentennial Term Associate Professor, Computer and Information Science
Sampath Kannan, Henry Salvatori Professor, Computer and Information Science
Michael Kearns, Professor and National Center Chair, Computer and Information Science
Qin Zhang, Assistant Professor, Computer Science Department, Indiana University Bloomington

Dedicated to my parents and my wife.

ACKNOWLEDGEMENT

It has truly been a rewarding and pleasant journey studying at Penn for the past six years. I have met so many great people during the process and received tremendous support from them. Here, I would like to take the opportunity to express my gratitude to everyone who has helped me along the way.

First of all, I want to thank my advisors Professor Sanjeev Khanna, Professor Boon Thau loo, and Professor Linh T. X. Phan. It is truly an honor and privilege advised by Sanjeev. I have learned so much from him over the years, especially regarding how to think with clarity, how to make progress when facing obstacles, how to properly explain thoughts or ideas, in terms of writing, presentation, or even just casually chatting, and the list goes on. Sanjeev made me a better student, a better collaborator, a better researcher, and most importantly, a better person, and I can never thank him enough for everything he did for me.

I was admitted to the PhD program at Penn because of Boon. To an international student who just finished undergraduate study (and with terrible English), the significance of this offer goes without saying. That alone is more than enough for me to thank Boon forever, not to mention the care and support that he provided me throughout my entire PhD life.

I want to thank Linh, along with all my co-authors: Sanchit Aggarwal, Sepehr Assadi, David Chiu, Tanveer Gill, Alexander J. T. Gurney, Andreas Haeberlen, Zachary Ives, Feifei Li, Changbin Liu, Bao-Liang Lu, David Maier, Suyog Mapara, Bart McManus, Victor M. Preciado, Yiqing Ren, Micah Sherr, Li-Chen Shi, Rui-Hua Sun, Val Tannen, Yufei Tao, Rakesh Vohra, Min Wang, Dong Xie, Bin Yao, Grigory Yaroslavtsev, Ke Yi, Zhuoyao Zhang, and Wenchao Zhou (names ordered alphabetically). It was a great pleasure working with them on many interesting projects over various domains, and I have learned a lot during the process. There are a few names among the list that I want to particularly show my appreciation. I want to thank Linh for helping me when I needed the most, providing me

with great advices on every aspect of the network functions virtualization project. Sepehr and I worked closely together for the majority of my PhD life. We learn and grow together as PhD students, and the success we had would not be possible without him. Feifei is the one who introduced to me the world of research. He and Ke taught me a lot about research when I was an intern at HP Labs China, and the paper we had back then played an extremely crucial role in my PhD application. In addition, Feifei also provided me with thorough support and great advices during the entire application process.

It is a great pleasure to have such a wonderful thesis committee formed by Professor Aaron Roth, Professor Sampath Kannan, Professor Michael Kearns, and Professor Qin Zhang. I want to thank the committee for the helpful comments they have provided me throughout the process. In particular, I want to thank Aaron for chairing the committee, Sampath and Michael for agreeing to be on the committee despite how packed their schedules are, and Qin for being my external and agreeing to join us remotely at such an unpleasantly early time in the morning.

I was a teaching assistant for Professor Susan Davidson on "Database and Information Science" and Professor Rajiv C. Gandhi on "Introduction to Algorithms". It was great experience working with them. As the head TA, I learned a lot from Susan on instructing a course, communicating with other TAs, and interacting with the students. Many interesting conversations happened when I was a TA for Rajiv, and our discussion about undergrad education in Penn were particularly inspiring to me.

Before coming to Penn, I was a student of ACM Honored Class in Shanghai Jiao Tong University. ACM Honored Class provided me with intense education that established my computer science foundation, and great opportunities to intern at HP Labs China and to be part of the BCMI Lab, mentored and supervised by Dr. Li-Chen Shi and Professor Bao-Liang Lu. These resource made it possible for me to pursuit my PhD and for that, I want to thank Professor Yong Yu both for creating this wonderful educational environment and for admitting me and allowing me to be a part of this family.

Life would be so boring without friends around. For that, I really need to thank Chen Chen for all the fun we had over the years, for all our interesting discussions on countlessly many different topics, and for being a great roommate. I also want to thank Zhepeng Yan for being a great friend and for improving my Chinese Chess skill significantly, thank Xiaoyuan Ma and Bo Shang for making me a better basketball player, and thank Lin Tan and Yupeng Lu for helping me pass through the first year at Penn, which is really the toughest among all. I have made many great friends during my stay at Penn, and it has been a great pleasure knowing all of you: Ling Ding, Dong Lin, Gang Song, Meng Xu, Yifei Yuan, Qizhen Zhang, Mingchen Zhao, Yumeng Zuo, and many others.

Last but not least, to my family, I am so fortunate to meet and fall in love with my beautiful and talented wife Qianru Jia, and I want to thank her for the support, the care, and the understanding, and for making our house a home. Finally, I want to express my deepest love and appreciation to my father Jie Li for guiding me through all crucial moments and my mother Qun Du for her unconditional love and care.

ABSTRACT

COMPUTING WITH DISTRIBUTED INFORMATION

Yang Li

Sanjeev Khanna

Boon Thau Loo, Linh T. X. Phan

The age of computing with massive data sets is highlighting new computational challenges. Nowadays, a typical server may not be able to store an entire data set, and thus data is often partitioned and stored on multiple servers in a distributed manner. A natural way of computing with such distributed data is to use distributed algorithms: these are algorithms where the participating parties (i.e., the servers holding portions of the data) collaboratively compute a function over the entire data set by sending (preferably small-size) messages to each other, where the computation performed at each participating party only relies on the data possessed by it and the messages received by it.

We study distributed algorithms focused on two key themes: convergence time and data summarization. Convergence time measures how quickly a distributed algorithm settles on a globally stable solution, and data summarization is the approach of creating a compact summary of the input data while retaining key information. The latter often leads to more efficient computation and communication. The main focus of this dissertation is on design and analysis of distributed algorithms for important problems in diverse application domains centering on the themes of convergence time and data summarization. Some of the problems we study include convergence time of double oral auction and interdomain routing, summarizing graphs for large-scale matching problems, and summarizing data for query processing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1 : Introduction

The term "Big Data" has quickly become one of the most popular keywords for the 21st century. A study showed that, as for the year 2014, every minute, Facebook users share nearly 2.5 million pieces of content, Twitter users tweet nearly $300,000$ times, and Instagram users post nearly $220,000$ new photos [4]. According to more recent studies, everyday, over 2.5 exabytes (which is 2.5 billion gigabytes) of data is generated all over the world [5], and Google alone needs to process over 20 terabytes (which is 20 million gigabytes) of these data [3].

The presence of such massive data sets leads to many technical challenges. One of the key challenges is how to efficiently store and access these data, as for data of this scale, storing them on a single server is often infeasible. To overcome this challenge, a natural idea is to *distribute* the data to multiple servers. This approach quickly becomes appealing and one of the key reasons is that for many applications, data are in fact originally generated in a distributed manner (e.g., think of data generated by Google users all over the world; naturally these data would be stored at different servers in different locations).

Another key challenge for handling massive data sets is how to efficiently process them and to extract useful information. In particular, with data stored in multiple locations in a distributed manner, efficiently processing these data (i.e., efficiently *computing with distributed information*) becomes even more challenging since traditional computation paradigm requires placing the entire dataset on a single server. One natural way of computing with distributed information is to use *distributed algorithms*: these are algorithms where the participating parties (i.e., the servers that hold portions of the entire data) collaboratively compute a function over the entire data set by sending (preferably small-size) messages to each other, where the computation performed at each participating party P only relies on the data possessed by P and the messages sent to P. The following two key concepts are closely connected to design and analysis of distributed algorithms.

Convergence. Using distributed algorithms, participating parties often follow the following pattern: computing based on their own data, sending/receiving messages to/from other parties, recomputing w.r.t. the received messages, sending/receiving new messages to/from other parties, and etc. For this pattern to be meaningful, a key property that a distributed algorithm must have is convergence (preferably rapid convergence: convergence in small number of steps). In other words, one needs to guarantee that after a certain number of steps, no party would have intention to send any new messages, and hence the algorithm terminates. Typically, a distributed algorithm needs to guarantee convergence no matter what the input data set is and how the data are distributed.

Data summarization. Another key concept of distributed algorithms (or of handling massive data sets in general) is *data summarization*: this is the approach of compressing or summarizing the input data into much smaller size while retaining key information. When designing distributed algorithms, a natural way of using the data summarization approach is to ask each party to summarize (i.e., to compress) her input data and the received messages into a small-size *sketch* and share the sketch with others.

The typical objective of the data summarization approach is to minimize the total size of the messages sent between different participating parties, i.e., to *minimize the total communication*. This line of research comes from the more general field of *sub-linear algorithms*: algorithms whose resource requirements are substantially smaller than the size of the input data. Designing distributed algorithms with minimum communication is generally referred to as the *distributed model of computation* for sub-linear algorithms.

When considering data summarization for distributed algorithms, we mostly focus on the goal of minimizing the total communication, with one exception where we using data summarization to pre-processing the input data before executing distributed algorithms with the goal of simplify the computation of the distributed algorithm.

This dissertation considers computation with distributed data. The main focus of this dis-

sertation is on design and analysis of distributed algorithms for diverse application domains centering around convergence and data summarization. Along the way, we investigate some of the important problems in these domains, and makes progress towards a better understanding of these problems.

1.1. Double Oral Auction

The concept of computing with distributed data was not formally introduced until late 1970s. However, the principle of performing computation in a distributed manner can be traced years before that. In particular, in early 1960s, Smith [102] designed a double oral auction (DOA) experiment for the following market trading problem.

There is a set of unit demand buyers (i.e., each buyer wants to buy an item) and a set of unit supply sellers (i.e., each seller has an item to sell), and all items are identical. Each buyer and each seller (i.e., each player) has his own valuation of an item, and the valuation may differ from player to player. For a buyer, his valuation is the highest price that he is willing to pay for an item, and for a seller, his valuation is the lowest price that he is willing to sell his item. The valuation of each player is his private information and naturally, players intend to keep their valuations private. Moreover, players behave selfishly: each buyer wants a buy an item for as cheap as possible and each seller wants to sell his item for as expensive as possible.

The goal of the market trading problem is to determine how the buyers and sellers should trade with each other and what should be the price for each trade, which we will refer to as an assignment of the market. In Smith's experiment, players found an assignment using the following simple double oral auction (DOA) mechanism. Buyers and sellers call out bids and offers where all other players can hear and decide whether or not to accept a bid or an offer. This continues until no more bids and offers are called out.

The DOA mechanism can be thought of as performing computation with distributed data as follows. The valuations of players (which are the entire data set) are distributed among

players and players communicate with each other through the bids and offers to compute a preferable assignment.

Remarkably, as noted by Smith, employing DOA, players consistently arrive at the Walrasian equilibrium, which, intuitively speaking, are assignments where every player is happy with the outcome and the market achieves maximum efficiency. As noted by Friedman [50], the outcome of Smith's experiment, is something of a mystery. How is it that the agents in the DOA overcome the impediments of both *private information* and *strategic uncertainty* to arrive at the Walrasian equilibrium? In the same survey [50], Friedman summarized various early theoretical attempts to explain this phenomenon, and concluded with a two-part conjecture: "First, that competitive (Walrasian) equilibrium coincides with ordinary (complete information) Nash Equilibrium (NE) in interesting environments for the DOA institution. Second, that the DOA promotes some plausible sort of learning process which eventually guides the *both clever and not-so-clever* traders to a behavior which constitutes an 'as-if' complete-information NE."

Over the years, the first part of Friedman's conjecture has been well studied and understood (see, e.g., [40]), but the second part of the conjecture is still awaiting proper explanation.

Contributions. We focus on giving a theoretical explanation to the second part of Friedman's conjecture. More specifically, we design a mechanism which simulates the DOA mechanism, and prove that this mechanism always converges to a Walrasian equilibrium in polynomially many steps. For our mechanism to convincingly simulates DOA, it captures the following four key properties of the DOA where all earlier attempts lacks in one or a few of them.

1. *Two-sided market:* Players on either side of the market can make actions.

2. *Private information:* When making actions, players have no other information besides their own valuations and the bids and offers submitted by others.

3. *Strategic uncertainty:* The players have the freedom to choose their actions modulo mild rationality conditions.

4. *Arbitrary recognition:* The auctioneer (only) recognizes bids and offers in an *arbitrary* order.

We further consider the more general scenario where not all players can hear and trade with each other. In other words, there is a network structure among the players, and a player can only communicate and trade with their neighbors on the network (the original setting of Smith's experiment can be viewed as the special case where the network is a complete bipartite graph). We show that DOA may not converge in this more general case, and a randomized variant of our mechanism (which can be viewed as a small modification of DOA) can be used to overcome this issue and still ensure convergence.

Our study on the convergence of the double oral auction mechanism along with its general bipartite graph variant was published in the 11th Conference on Web and Internet Economics (WINE), 2015 [15] (best paper award).

1.2. Stochastic Matching

In many trading games, data are naturally distributed since players naturally have private information. The market setting for double oral auction is a simple and classic example where the private information of each player is just their valuation of an item. The following is another interesting model for a simple and classic trading game.

Suppose there is a collection of players where each player wants to find another player to trade. Each player P is given a list of other players that P can trade with, and P can propose to trade with any player on his list. However, any player on the list may reject P's proposal with some probability. The goal is for each player to make only a few trade proposal while ensuring that among the accepted proposals, a large number of player pairs can trade simultaneously.

The above trading game can be formulated as a graph problem as follows. Create a node for each player and the neighborhood (i.e., the incident edges) of a player P is the list of other players that P can trade with. Each edge may be removed with some probability (i.e., the proposal gets rejected), and we say an edge is *realized* if it is not removed (i.e., the proposal gets accepted). The goal is to identify a bounded degree subgraph where with high probability, among the set of realized edges in the subgraph, there is a matching of large size.

Note that for this problem, we are using the data summarization approach to reduce the size of the trading graph (i.e., to pre-processing the input data). After preprocessing, the number of neighbors that each player has would be bounded, and hence when the players use (any) distributed algorithm to find a trading assignment (or a matching), the computation of the distributed algorithm would be simplified.

This graph problem is formally known as the *stochastic matching* problem [25], which has received great attention mainly due to the following application in kidney exchange. Often patients who need a kidney transplant have a family member who is willing to donate his/her kidney, but this kidney might not be a suitable match for the patient due to reasons like incompatible blood-type etc. To solve this problem, a kidney exchange is performed in which patients *swap* their incompatible donors to each get a compatible donor.

In terms of the above trading game, each patient-donor pair is a player (or a node) and the neighbor of a patient-donor pair is the set of all other pairs where a kidney exchange is possible. The goal is to find a large set of exchange (i.e., a large matching). The reason that a trade (or an exchange) proposal might be rejected is that when identifying the neighbors of a patient-donor pair, typically one only has access to the medical record of the patients and the donors, which can be used to rule out the patient-donor pairs where donation is impossible (e.g., different blood types), but does not provide a conclusive answer for whether or not a donation is indeed feasible. Therefore, after proposed one needs to run more (potentially costly) tests and with some probability, one or both of the donation might

be infeasible, which would make the exchange itself infeasible.

Contributions. We focus on solving the stochastic matching problem for two generally considered settings: adaptive and non-adaptive. In the adaptive setting, one can identify the target low-degree subgraph step by step while the computation is in each step can be based on the outcome of previous steps (i.e., which of the previously used edges are realized). In the non-adaptive setting, one needs to output a low-degree subgraph in one shot where guaranteeing that a large matching would be realized from this subgraph.

We design both adaptive and non-adaptive algorithms for the stochastic matching problem which achieves the same approximation ratio as the previous best bound [25]. However, the degrees of the subgraph that our algorithms output are not only exponentially smaller than the state of the art, but also essentially the best possible.

Our work on the stochastic matching problem was published in the 17th ACM Conference on Economics and Computation (EC), 2016 [13].

1.3. Convergence Time of Interdomain Routing

In the decade following Smith's double oral auction experiment, the field of computer networking started to flourish. This is the time when several amazing accomplishments were made and the Internet started to come into our lives. Consequently, these accomplishments also make 1970s a landmark, or the start, for computing with distributed data: the whole idea of storing data (and process them) in a distributed manner relies on the ability of transmitting data between any two computers, which was only made possible by the success of the Internet.

The Internet is a network of networks, formed by many Autonomous Systems (ASes). Each AS could be a telecommunication company, a university, etc. Some ASes are directly connected to each other and some ASes need to go through a sequence of other ASes to form a connection. To ensure connection between any two computers on the Internet, it

is necessary that any two ASes are connected. With the presence of many possible routes between ASes, specific routes needs to be chosen and agreed by all ASes. The process where ASes determine preferable routes to connect to each another is called interdomain routing, and the current protocol for interdomain routing is BGP, Border Gateway Protocol [97].

The most important difference between interdomain routing and other route-finding methods is the way in which local policy controls the selection and propagation of routes: rather than there being a single global definition of 'best' route (as in shortest-path protocols), every AS has its own interpretation. This is because of the asymmetry in the economic relationships among these network participants; accordingly, BGP is best perceived as a protocol that tries to compute a Nash equilibrium in a game where all parties are trying to obtain good routes (by their local definition) but are constrained by the choices of others (one cannot pick a route through a neighbor without agreement) [93, 46].

Therefore, interdomain routing is naturally a computational problem with distributed information as follows. The preferences of all ASes along with the network topology (which are the entire data set) are distributed among ASes and the goal is to determine routes between any two ASes which is consistent with the preferences, which we will refer to as stable routes.

It is well known that in general, network policies can conflict to the extent that BGP will be unable to find stable routes: in this case, the protocol will *oscillate* indefinitely [81, 80, 88]. There has been a great deal of work on identifying sufficient criteria for BGP to converge to a unique stable state [60, 62, 64, 107, 52, 104] using abstract models (in particular, *stable paths problems [63]*). On the other hand, it was observed that even if BGP *does* converge, it may take some time to do so [80, 47]. The slow convergence causes practical difficulties for network operators, and degradation or loss of service for their customers.

Contributions. We focus on the theoretical aspect of the convergence *time* of BGP. In particular, we aim to understand *when eventual convergence is guaranteed*, how much time

it requires for the network to actually converge (or to stabilize). We study convergence time when different restrictions are applied on the structure of the preferences, and establish both necessary and sufficient conditions (i.e., dichotomy theorems) for convergence in *polynomial time*. To the best of our knowledge, prior to our work, the only related studies concerning polynomial time convergence are given by [52, 99], which only established polynomial time convergence for the Gao-Rexford criteria.

The key challenge of our work lies in the fact that ASes might act (i.e., change their routes) in any arbitrary order. In other words, we need to understand how bad the arbitrary activation order can hurt convergence time and also, how to establish maximum convergence time with the presence of arbitrary activation.

We further observe that given the preferences of the ASes, many different stable routes are possible. However, other than being stable, some routes could be considered better for other objectives. This leads to the question of designing efficient algorithms that add only a few more preferences (which is consistent with the original preferences) such that the resulting stable routes has desired properties, such as the length of the longest path in the resulting stable routes is minimized. We provide both hardness results and approximation algorithms for this problem.

Our study on understanding BGP convergence time and the problem of obtaining stable routes with desired properties was published in IEEE International Conference on Computer Communications (INFOCOM), 2016 [68].

1.4. Maximum Matching with Minimum Communication

During the past several decades and continuing on to recent years, there has been tremendous research on distributed algorithms for graph problems [110, 45, 10, 71, 69]. The maximum matching problem, one of the most well-studied problems in classical optimization, has been extensively studied in the distributed model due to its central role in many applications [84].

The distributed setting for the matching problem can be described as follows. The input graph is *adversarially* partitioned across k parties and the goal is to design distributed algorithms (or generally referred to as *protocols*) such that the k parties can jointly compute a maximum matching. Two natural ways of partitioning the input graph have been considered: in the *edge partition model*, each party holds a subset of edges of an input (possibly multi-) graph while in the *vertex partition model* the input graph must be bipartite and each player holds a distinct subset of *vertices* on the left together with all their adjacent edges. The performance of the protocol is typically measured by the *total communication*, which is simply the total size of all messages sent/received by every parties.

Many variations (especially regarding the communication model) of the aforementioned distributed setting has been studied. For instance, Huang *et al.* [71] focused on the k-party *message-passing* model with two-way player-to-player communication. This is the model where any party can send messages to any other party in any order. This is probably the most general communication model for distributed computation since there is essentially no restrictions on who can communicate with whom or whether who should send messages earlier than others. For this model, Huang *et al.* [71] showed that the communication of $\Theta\left(\frac{nk}{\alpha^2}\right)$ bits is both necessary and sufficient for computing an α-approximate matching for both vertex partition and edge partition models; here n denotes the number of vertices in the input graph.

The work of Dobzinski *et al.* [45, 10] considers another distributed computation model where the input graph is bipartite and each party possesses a distinct vertex on the left along with all edges incident on this vertex. This is a special setting for the vertex partition model where the number of parties equals the number of vertices on the left. Dobzinski *et al.* [45] focuses on simultaneous communication (or simultaneous protocols) where every party simultaneously sends a message to a *coordinator*, and the coordinator computes a large matching using the messages. Dobzinski *et al.* [45] showed that for the simultaneous protocol where each party only sends a random incident edge to the coordinator, the coor-

dinator can only output an $O(\sqrt{n})$-approximate matching, which matches the lower bound of [71].

Contributions. We mainly focus on simultaneous protocols for the k-party vertex partition model, where there are k parties and each party holds a distinct set of vertices on the left along with all incident edges; each party simultaneously sends a message to the coordinator, and the coordinator computes a large matching using all received messages. This model generalizes the model studied by Dobzinski *et al.* [45] (which is the case where $k = n$). We show that for achieving an α-approximation, $\widetilde{O}(\frac{nk}{\alpha^2})$ bits of communication is still sufficient for this more restricted communication model (compared to Huang *et al.* [71]). In particular, our result implies that the lower bound of $\Omega(\frac{nk}{\alpha^2})$ bits of communication (of Huang *et al.* [71]) is not only tight for the more general two-way player-to-player communication model, but also tight for the more restricted simultaneous communication model.

Moreover, we should also point out that the setting for the double oral auction (DOA) experiment (Section 1.1) is also a distributed setting for the (weighted) maximum matching problem: the weight of an edge between a buyer and a seller equals the valuation of the buyer minus the valuation of the seller (i.e., the weights of the edges are vertex induced), and DOA is in fact a distributed protocol that allows the players to compute a maximal weighted matching in the input graph. Therefore, our work on understanding DOA also contributes to understanding matching in distributed settings.

Our work on designing simultaneous protocols that compute maximum matching for the vertex-partition model was published in the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), 2016 [16].

1.5. Data Provisioning

During the study of computation with distributed information, many interesting models are introduced. For instance, for the aforementioned market trading and interdomain routing problems, the goal is to perform distributed computation with private information, while

ensuring rapid convergence and/or converging to desired states. For the aforementioned maximum matching problem, the goal is to perform distributed computation with minimum communication, while computing a pre-specified function. With more interest placed on computation with distributed data, more applications emerge which leads to more interesting models [76, 82, 77, 89]. Here is one of these models that we recently introduced and studied.

Suppose the entire dataset is collected from k different sources, and the data of different sources resides on different machines. When performing data analytics, often some sources are considered less trustworthy than others. Moreover, whether or not a source can be trusted may depend on many factors which may be subject to change (e.g., the credibility of the company that collects the data, the business relationship between the companies, etc). Therefore, an analyst might be interested in considering different subsets of the sources, combining the data collected by these sources, and perform certain analysis.

A simple solution to handle this issue of combining different subsets of sources is to collect all data from all sources to a single location. Then, whenever the analyst wants to consider some subset of sources, he can directly combine the corresponding data and perform the required computation. However, since the dataset collected by each source could be massive, there are two issues of this approach: it might be infeasible to place all data to a single machine, and performing (potentially costly) computation on such large dataset over and over (i.e., whenever a source becomes/is no longer trusted) could be rather computationally inefficient.

The overcome these issues, we introduce the following computation model for distributed data referred to as provisioning. Each source only sends a sketch to the analyst such that for any subset of the sources, the answer of the designated analytical question can be computed using only the sketches. The goal is to compute and send sketches with small size (preferably exponentially smaller than the input size).

Notice that if we further constrain the model and require each source to directly send the sketch to the analyst simultaneously, this model essentially degrades to the typical distributed computation model with simultaneous protocols: given any simultaneous protocol Π of the distributed computation model, design an algorithm for provisioning as follows: each source runs the protocol Π as if it is trusted and sends the message (which will be the sketch for provisioning) to the analyst; whenever the analyst wants to analyze for a subset of sources, he runs Π only over the sketches sent from trusted sources (and ignore the other sketches). The correctness of this algorithm is guaranteed by the correctness of the protocol Π. If Π is randomized, one only need to boost the probability of success accordingly, which in all applications that we considered, only leads a blowup of factor k.

Contributions. We design provisioning algorithms for database Select-Project-Join (SPJ) queries with standard aggregation functions including count, sum, average, and quantiles. The sketch size of our algorithms are all $\text{poly}(\log n, k)$, where n is the size of the entire dataset, and k is the number of sources. Moreover, the algorithms are also time efficient in the sense that the running time for computing and sending the sketches is $O(n) + \text{poly}(\log n, k)$, and the running time for computing the answer of any subset of sources is $\text{poly}(\log n, k)$. Notice that using the provisioning model, the total running time for computing the answer of all subsets of sources is $2^k \cdot \text{poly}(\log n, k)$, while for the aforementioned simple solution where each source sending all data, reading the data for each subset of sources already takes time $2^k \cdot n$.

Our study on designing algorithms for the provisioning model was published in 19th International Conference on Database Theory (ICDT), 2016 [14].

CHAPTER 2 : Double Oral Auction

Starting from this chapter, we explain in more details the problems centering around distributed information that we studied, along with the methodologies that we developed. We begin with two simple trading games that we mentioned in the previous section, namely the double oral auction and the stochastic matching problem.

In particular, in this chapter, we will focus on the convergence property of double oral auction (DOA)[1]. We will first formalize the buyer-seller trading market used in the DOA experiment and design a mechanism for this market that simulates DOA. Then, we prove that this mechanism always converges to a *Walrasian equilibrium* in polynomially many steps, which proves the second part of Friedman's conjecture. Finally, we further consider the more general scenario where not all players can trade with each other, and show that DOA may not converge in this more general case. To overcome this issue, we propose a randomized variant of our mechanism (which can be viewed as a minor modification of DOA) which still ensure convergence. We start with a short introduction on the history of DOA.

2.1. Background

The study on double oral auction started from a market experiment conducted by Chamberlin in which prices failed to converge to a Walrasian equilibrium [28]. Chamberlin's market was an instance of the assignment model with homogeneous goods. There is a set of unit demand buyers and a set of unit supply sellers, and all items are identical. Each agent's value or opportunity cost for the good is their private information and preferences are quasi-linear. Chamberlin concluded that his results showed competitive theory to be inadequate. Vernon Smith, in an instance of insomnia, recounted in [103] demurred:
"The thought occurred to me that the idea of doing an experiment was right, but what was wrong was that if you were going to show that competitive equilibrium was not re-

[1]The full paper of this work can be find in [15].

alizable ... you should choose an institution of exchange that might be more favorable to yielding competitive equilibrium. Then when such an equilibrium failed to be approached, you would have a powerful result. This led to two ideas: (1) why not use the double oral auction procedure, used on the stock and commodity exchanges? (2) why not conduct the experiment in a sequence of trading days in which supply and demand were renewed to yield functions that were daily flows?"

Instead of Chamberlin's unstructured design, Smith used a different design, a double oral auction (DOA) scheme, in which both buyers and sellers call out bids or offers which an auctioneer recognizes [102]. Transactions resulting from accepted bids and offers are recorded. This continues until there are no more acceptable bids or offers. At the conclusion of trading, the trades are erased, and the market reopens with valuations and opportunity costs unchanged. The only thing that has changed is that market participants have observed the outcomes of the previous days trading and may adjust their expectations accordingly. This procedure was iterated four or five times. Smith was astounded: "I am still recovering from the shock of the experimental results. The outcome was unbelievably consistent with competitive price theory." [103](p. 156)

As noted by Daniel Friedman [50], the results in [102], replicated many times, are something of a mystery. How is it that the agents in the DOA overcome the impediments of both *private information* and *strategic uncertainty* to arrive at the Walrasian equilibrium? A brief survey of the various (early) theoretical attempts to do so can be found in Chapter 1 of [50]. Friedman concluded his survey of the theoretical literature with a two-part conjecture. "First, that competitive (Walrasian) equilibrium coincides with ordinary (complete information) Nash Equilibrium (NE) in interesting environments for the DOA institution. Second, that the DOA promotes some plausible sort of learning process which eventually guides the *both clever and not-so-clever* traders to a behavior which constitutes an 'as-if' complete-information NE."

Over the years, the first part of Friedman's conjecture has been well studied (see, e.g., [40];

see also Section 2.4) but the second part of the conjecture is still left without a satisfying resolution. The focus of this paper is on the second part of Friedman's conjecture. More specifically, we design a mechanism which simulates the DOA, and prove that this mechanism always converges to a Walrasian equilibrium in polynomially many steps. Our mechanism captures the following four key properties of the DOA.

1. *Two-sided market:* Agents on either side of the market can make actions.

2. *Private information:* When making actions, agents have no other information besides their own valuations and the bids and offers submitted by others.

3. *Strategic uncertainty:* The agents have the freedom to choose their actions modulo mild rationality conditions.

4. *Arbitrary recognition:* The auctioneer (only) recognizes bids and offers in an *arbitrary* order.

Among these four properties, mechanisms that allow agents on either side to make actions (*two-sided market*) and/or limit the information each agent has (*private information*) have received more attention in the literature (see Section 2.2). However, very little is known for mechanisms that both work for strategically uncertain agents and recognize agents in an arbitrary order. Note that apart from resolving the second part of Friedman's conjecture, having a mechanism with these four properties itself is of great interest for multiple reasons. First, in reality, the agents are typically unwilling to share their private information to other agents or the auctioneer. Second, agents naturally prefer to act freely as oppose to being given a procedure and merely following it. Third, in large scale distributed settings, it is not always possible to find a real auctioneer who is trusted by every agent, and is capable of performing massive computation on the data collected from all agents. In the DOA (or in our mechanism) however, the auctioneer only recognizes actions in an arbitrary order, which can be replaced by any standard distributed token passing protocol, where an agent can take an action only when he is holding the token. In other words, our mechanism serves

more like a platform (rather than a specific protocol) where rational agents always reach a Walrasian equilibrium no matter their actual strategy. To the best of our knowledge, no previous mechanism enables such a 'platform-like' feature.

2.2. Our Results and Related Work

We design a mechanism that simulates DOA by simultaneously capturing two-sided market, private information, strategic uncertainty, and arbitrary recognition. More specifically, following DOA, at each iteration of our mechanism, the auctioneer maintains a list of active price submission and a tentative assignment of buyers to sellers that 'clears' the market at the current prices (note that this can also be distributedly maintained by the agents themselves). Among the agents who wish to make or revise an earlier submission, an arbitrary one is recognized by the auctioneer and a new tentative assignment is formed. An agent can submit *any* price that strictly improves his payoff given the current submissions (rather than being forced to make a 'best' response, which is to submit the price that maximizes payoff). We show that as long as agents make myopically better responses, the market always converges to a Walrasian equilibrium in polynomial number of steps. Furthermore, *every* Walrasian equilibrium is the limit of some sequence of better responses. We should remark that the fact that an agent always improves his payoff does not imply that the total payoff of all agents always increases. For instance, a buyer can increase his payoff by submitting a higher price and 'stealing' the current match of some other buyer (whose payoff would drop).

To the best of our knowledge, no existing mechanism captures all four properties for the DOA that we proposed in this paper. For most of the early work on auction based algorithms (e.g., [101, 79, 39, 40, 22]), unlike the DOA, only one side of the market can make offers. By permitting only one side of the market to make offers, the auction methods essentially pick the Walrasian equilibrium (equilibria are not unique) that maximizes the total surplus of the side making the offers.

For two-sided auction based algorithms [24, 23], along with the 'learning' based algorithms studied more recently [91, 74], agents are required to follow a specific algorithm (or protocol) that determines their actions (and hence violates strategic uncertainty). For example, [24] requires that when an agent is activated, a buyer always matches to the 'best' seller and a seller always matches to the 'best' buyer (i.e., agents only make myopically *best* responses, which is not the case for the DOA). [74] has agents submit bids based on their current best alternative offer and prices are updated according to a common formula relying on knowledge of the agents opportunity costs and marginal values. [91], though not requiring agents to always make myopically best responses, has agents follow a specific (randomized) algorithm to submit conditional bids and choose matches. We should emphasize that agents acting based on some *random* process is different from agents being strategically uncertain. In particular, for the participants of the original DOA experiment (of [102]), there is no a priori reason to believe that they were following some specific random procedure during the experiment. On the contrary, as stated in Friedman's conjecture, there are *clever and not-so-clever* participants, and hence different agents could have completely different strategies and their strategies might even change when, for instance, seeing more agents matching with each other, or by observing the strategies of other agents. Therefore, analyzing a process where agents are strategically uncertain can be distinctly more complex than analyzing the case where agents behave in accordance with a well-defined stochastic process. In this paper, we consider an extremely general model of the agents: the agents are acting arbitrarily while only following some mild rationality conditions. Indeed, proving fast convergence (or even just convergence) for a mechanism with agents that are strategically uncertain is one of the main challenges of this work.

Arbitrary recognition is another critical challenge for designing our mechanism. For example, the work of [95, 91] deploys randomization in the process of recognizing agents. This is again in contrast to the original DOA experiment, since the auctioneer did not use a randomized procedure when recognizing actions, and it is unlikely that the participants *decide* to make an action following some random process (in fact, some participants might

be more 'active' than others, which could lead to the 'quieter' participants barely getting *any* chance to make actions, as long as the 'active' agents are still making actions).

The classical work on the *stable matching* problem [51] serves as a very good illustration for the importance of arbitrary recognition. Knuth [78] proposed the following algorithm for finding a stable matching. Start with an arbitrary matching; if it is stable, stop; otherwise, pick a blocking pair and match them; repeat this process until a stable matching is found. Knuth showed that the algorithm could cycle if the blocking pair is picked *arbitrarily*. Later, [98] showed that picking the blocking pairs at random suffices to ensure that the algorithm eventually converges to a stable matching, which suggests that it is the arbitrary selection of blocking pairs that causes Knuth's algorithm to not converge.

The setting of Knuth's algorithm is very similar to the process of the DOA in the sense that in any step of the DOA, a temporary matching is maintained and agents can make actions to (possibly) change the current matching. But perhaps surprisingly, we show that arbitrary recognition does not cause the DOA to suffer from the same cycling problem as Knuth's algorithm. The main reason, or the main difference between the two models is that our assignment model involves both matching and prices, while Knuth's algorithm only involves matchings. As a consequence, in our mechanism, the preferences of the agents change over time (since an agent always favors the better price submission, the preferences could change when new prices are submitted). In the instance that leads Knuth's algorithm to cycle (see [78]), the fundamental cause is that the preferences of *all* agents form a cycle. However, in our mechanism, preferences (though changing) are always consistent for all agents.

Based on this observation, we establish the limit of the DOA by introducing a small friction into the market: restricting the set of agents on the other side that each agent can trade with[2]. We show that in this case, there is an instance with a specific adversarial order of

[2]In Chamberlin's experiment, buyers and sellers had to seek each other out to determine prices. This search cost meant that each agent was not necessarily aware of all prices on the other side of the market.

recognizing agents such that following this order, the preferences of the agents (over the entire order) form a cycle and the DOA may never converge. Finally, we complete the story by showing that if we change the mechanism to recognize agents randomly, with high probability, a Walrasian equilibrium will be reached in polynomial number of steps. This further emphasizes the distinction between random recognition and arbitrary recognition for DOA-like mechanisms.

Organization: The rest of this chapter is organized as follows. In Section 2.3, we formally introduce the model of the market and develop some preliminaries. Section 2.4 establishes a connection between the stable states of the market and social welfare. Our main results are presented in Section 2.5. We describe our DOA style mechanism and show that in markets with no trading restrictions, it converges in a number of steps that is polynomially bounded in the number of agents. We then show that when each agent is restricted to trade only with an arbitrary subset of agents, the mechanism need not converge. A randomized variant of our mechanism is then presented to overcome this issue. Finally, we conclude with some directions for future work in Section 2.6.

2.3. Preliminaries

We will use the terms 'player' and 'agent' interchangeably throughout the paper. We use B to represent a buyer, S for a seller, and Z for either of them (i.e., a player). Also, b is used as the bid submitted by a buyer and s as the offer from a seller.

Definition 2.1 (Market)**.** *A market is denoted by $G(\mathcal{B}, \mathcal{S}, E, val)$, where $\mathcal{B}$ and $\mathcal{S}$ are the sets of buyers and sellers, respectively. Each buyer $B \in \mathcal{B}$ is endowed with a valuation of the item, and each seller $S \in \mathcal{S}$ has an opportunity cost for the item. We slightly abuse the terminology and refer to both of these values as the valuation of the agent for the item. The valuation of any agent Z is chosen from range $[0, 1]$, and denoted by $val(Z)$. Finally, E is the set of undirected edges between $\mathcal{B}$ and $\mathcal{S}$, which determines the buyer-seller pairs that may trade.*

Let $m = |E|$ and $n = |\mathcal{B}| + |\mathcal{S}|$.

Definition 2.2 (Market State)**.** *The* state *of a market at time t is denoted $\mathcal{S}^t(P^t, \Pi^t)$ ($\mathcal{S}(P, \Pi)$ for short, if time is clear or not relevant), where P is a* price function *revealing the price submission of each player and Π is a* matching *between $\mathcal{B}$ and $\mathcal{S}$, indicating which players are currently paired. In other words, the bid (offer) of a buyer B (seller S) is $P(B)$ ($P(S)$), and B, S are paired in Π iff $(B, S) \in \Pi$. In addition, we denote a player $Z \in \Pi$ iff Z is matched with some other player in Π, and denote his match by $\Pi(Z)$.*

Furthermore, the state where each buyer submits a bid of 0, each seller submits an offer of 1, and no player is matched is called the zero-information state*.*

We use the term zero-information because no player reveals non-trivial information about his valuation in this state.

Definition 2.3 (Valid State)**.** *A state is called* valid *iff (a_1) the price submitted by each buyer (seller) is lower (higher) than his valuation, (a_2) two players are matched only when there is an edge between them, and (a_3) for any pair in the matching, the bid of the buyer is no smaller than the offer of the seller.*

In the following, we restrict attention to states that are valid.

Definition 2.4 (Utility)**.** *For a market $G(\mathcal{B}, \mathcal{S}, E, val)$ at state $\mathcal{S}(P, \Pi)$, the* utility *of a buyer is defined as $val(B) - P(B)$, if B receives an item, and zero otherwise. Similarly, the utility of a seller is defined as $P(S) - val(S)$, if S trades his item, and zero otherwise.*

Note that what we have called utility is also called surplus.

Definition 2.5 (Stable State)**.** *A* stable state *of a market $G(\mathcal{B}, \mathcal{S}, E, val)$ is a state $\mathcal{S}(P, \Pi)$ s.t. (a_1) for all $(B, S, \in)E$, $P(B) \leq P(S)$ (a_2) if $Z \notin \Pi$, then $P(Z) = val(Z)$, and (a_3) if $(B, S) \in \Pi$, then $P(B) = P(S)$.*

Suppose $\mathcal{S}(P, \Pi)$ is not stable. Then, one of the following must be true.

1. There exists $(B, S) \in E$ such that $P(B) > P(S)$. Then, both B and S could strictly

increase their utility by trading with each other using the average of their prices.

2. There exists $Z \notin \Pi$ such that $P(Z) \neq val(Z)$. This agent could raise his bid (if a buyer) or lower his offer (if a seller), without reducing his utility and having a better opportunity to trade.

3. There exists $(B, S) \in \Pi$ such that $P(B) > P(S)$ ($P(B) < P(S)$ results in an invalid state). One of the agents could do better by either raising his offer or lowering his bid.

Definition 2.6 (ϵ-Stable State)**.** *For any $\epsilon \geq 0$, a state $\mathcal{S}(P, \Pi)$ of a market $G(\mathcal{B}, \mathcal{S}, E, val)$ is ϵ-stable iff (a_1) for any $(B, S, \in)E$, $P(B) - P(S) \leq \epsilon$ (a_2) if player $Z \notin \Pi$, $P(Z) = val(Z)$, and (a_3) if $(B, S) \in \Pi$, $P(B) = P(S)$.*

Note that the only difference between a stable state and an ϵ-stable state lies in the first property. At any ϵ-stable state, no matched player will have a move to increase his utility by more than ϵ.

Definition 2.7 (Social Welfare)**.** *For a market $G(\mathcal{B}, \mathcal{S}, E, val)$ with a matching Π, the social welfare (SW) of this matching is defined as the sum of the valuation of the matched buyers minus the total opportunity cost of the matched sellers. We denote by $\mathtt{SW}_\Pi$ the SW of matching Π.*

Definition 2.8 (ϵ-approximate SW)**.** *For any market, a matching Π is said to give an ϵ-approximate SW if $\mathtt{SW}_\Pi \geq \mathtt{SW}_{\Pi^*} - n\epsilon$ for any Π^* that maximizes SW. In other words, on average, the social welfare collected from each player using Π is at most ϵ less than that collected using Π^*.*

2.4. Stable State and Social Welfare

In this section we mainly establish the connection between stables states and social welfare in the market. We emphasize that most results in this section are well known in the literature and stated here for the sake of completeness.

The problem of finding a matching that maximizes SW can be formulated as a linear program (LP) (see [23] for example). For any edge $(B, S, \in)E$, let $x_{B,S}$ be the variable indicating whether $(B, S, \ )$is selected in the matching or not, and define weight of the edge, $w_{B,S} = \text{val}(B) - \text{val}(S)$. Therefore, the LP (primal) and its dual can be defined as follows.

$$\begin{aligned}
&\max \sum_{(B,S,\in)E} w_{B,S} \cdot x_{B,S} && \min \sum_{B\in\mathcal{B}} y_B + \sum_{S\in\mathcal{S}} y_S \\
&\text{s.t. } \forall B^* \in \mathcal{B}, \sum_{(B^*,S,\in)E} x_{B^*,S} \leq 1 && \text{s.t. } \forall (B, S, \in)E, y_B + y_S \geq w_{B,S} \\
&\quad \forall S^* \in \mathcal{S}, \sum_{(B,S^*,\in)E} x_{B,S^*} \leq 1 && \quad y_B, y_S \geq 0 \\
&\quad x_{B,S} \geq 0
\end{aligned}$$

In the following, we will refer to the above linear programs as 'primal' and 'dual', respectively. The dual variables y can be interpreted as the utilities that agents enjoy assuming every buyer gets an item and every seller sells the item. Since it only depends on the price function, we call this price-wise utility. The constraint $y_B + y_S \geq w_{B,S}$ essentially states that the sum of the utilities obtained by (B, S) must be at least as large as their gains from trade.

We use x^Π to denote the characteristic function of matching Π, i.e., $x^\Pi_{B,S} = 1$ iff $(B, S) \in \Pi$, and use y^P to denote the price-wise utility function of a price function P, i.e., $y^P_B = \text{val}(B) - P(B)$ and $y^P_S = P(S) - \text{val}(S)$. It is well known that SW is maximized at a Walrasian equilibrium (see [23]) and we state here a similar result for stable states.

Theorem 2.1. *A state $\mathcal{S}(P, \Pi)$ is stable iff x^Π is an optimal solution for the primal and y^P is an optimal solution for the dual.*

Proof. To see the forward direction, assume $\mathcal{S}(P, \Pi)$ is a stable state. We first verify that x^Π and y^P are indeed feasible solutions. x^Π is clearly feasible since it is characteristic function of a valid matching. y^P preserves non-negativity constraint of dual, since no

player could submit a price exceeding his valuation in P. Moreover, we can write $y_B^P + y_S^P = \text{val}(B) - P(B) + P(S) - \text{val}(S) = w_{B,S} + P(S) - P(B)$. By property (a_1) of stable states, $P(S) \geq P(B)$, hence $y_B^P + y_S^P \geq w_{B,S}$, preserving the dual constraint and implying that y^P is also feasible.

To prove optimality of x^Π and y^P, using weak duality, we only need to verify that value of primal is equal to value of dual.

$$\sum_{B \in \mathcal{B}} y_B + \sum_{S \in \mathcal{S}} y_S = \sum_{(B,S) \in \Pi} (\text{val}(B) - P(B) + P(S) - \text{val}(S)) + \sum_{Z \notin \Pi} y_Z^P \tag{2.1}$$

$$= \sum_{(B,S) \in \Pi} (\text{val}(B) - \text{val}(S)) = \sum_{(B,S,\in) E} w_{B,S} \times x_{B,S}^\Pi \tag{2.2}$$

(1) to (2) uses properties (a_1) and (a_2) of stable states, $P(B) = P(S)$ for $(B, S) \in \Pi$, and $P(Z) = \text{val}(Z)$ for $Z \notin \Pi$. Thus x^Π and y^P are optimal solutions of primal and dual, respectively.

For the reverse direction, assume (x^Π, y^P) is a pair of optimal primal and dual solutions . Since y^P is a feasible solution, as we just stated, $y_B^P + y_S^P \geq w_{B,S}$ will give us $P(S) \geq P(B)$, thus property (a_1) of stable states holds. For properties (a_2) and (a_3) of stable states, since

$$\sum_{B \in \mathcal{B}} y_B + \sum_{S \in \mathcal{S}} y_S = \sum_{(B,S,\in) E} w_{B,S} \times x_{B,S}^\Pi$$

$$\sum_{(B,S) \in \Pi} (\text{val}(B) - P(B) + P(S) - \text{val}(S)) + \sum_{Z \notin \Pi} y_Z^P = \sum_{(B,S) \in \Pi} \text{val}(B) - \text{val}(S)$$

$$\sum_{(B,S) \in \Pi} (P(S) - P(B)) + \sum_{Z \notin \Pi} y_Z^P = 0$$

Since $P(S) \geq P(B)$ and also y^P is a non-negative vector, both terms in the last expression must be zero, which implies that properties (a_2) and (a_3) of stable states also hold. Therefore $\mathcal{S}(P, \Pi)$ is a stable state. □

Theorem 2.1 states that any stable state maximizes SW. In other words, a stable state is a Walrasian equilibrium of the market. Moreover, *any* pair of optimal primal and dual solutions can form a stable state. We now show that for a sufficiently small ϵ, an ϵ-stable state is almost as good as stable states in terms of achieving maximum SW.

Theorem 2.2. *For any market $G(\mathcal{B}, \mathcal{S}, E, val)$, for any $\epsilon > 0$, any ϵ-stable state realizes an ϵ-approximate SW. Moreover, for*

$$\delta = \min \{|val(Z_1) - val(Z_2)| \mid Z_1, Z_2 \in \mathcal{B} \cup \mathcal{S}, val(Z_1) \neq val(Z_2)\},$$

we have for $0 \leq \epsilon < \delta/n$, any ϵ-stable state maximizes SW.

To simplify the notation, we treat an agent who is unmatched as being matched with themselves. To this end, for each buyer we introduce a *dummy* seller with an opportunity cost equal to his valuation, similarly for each seller. An agent matched with their dummy counterpart is interpreted as being unmatched. We denote the dummy seller of buyer B as $\overline{S}_B$ and the dummy buyer of seller S as $\overline{B}_S$.

Proof. We define the following ϵ-primal and ϵ-dual pair.

$$\begin{aligned}
&\max \sum_{(B,S,\in)E} (w_{B,S} - \epsilon) x_{B,S} && \min \sum_{B \in \mathcal{B}} y_B + \sum_{S \in \mathcal{S}} y_S \\
&\text{s.t. } \forall B^* \in \mathcal{B}, \sum_{(B^*,S,\in)E} x_{B^*,S} \leq 1 && \text{s.t. } \forall (B,S,\in)E, y_B + y_S \geq (w_{B,S} - \epsilon) \\
&\quad \forall S^* \in \mathcal{S}, \sum_{(B,S^*,\in)E} x_{B,S^*} \leq 1 && \quad y_B, y_S \geq 0 \\
&\quad x_{B,S} \geq 0
\end{aligned}$$

Given a ϵ-stable state $\mathcal{S}(P, \Pi)$, since property (a_1) of ϵ-stable states is equivalent to $\text{val}(B) - P(B) + P(S) - \text{val}(S) \geq w_{B,S} - \epsilon$, y^P is a feasible solution of ϵ-dual (non-negativity constrains

hold since P is valid price function). By properties (a_2) and (a_3) of ϵ-stable states,

$$\begin{aligned}\sum_{B\in\mathcal{B}} y_B^P + \sum_{S\in\mathcal{S}} y_S^P &= \sum_{(B,S)\in\Pi} (\text{val}(B) - P(B) + P(S) - \text{val}(S)) + \sum_{Z\notin\Pi} y_Z^P \\ &= \sum_{(B,S)\in\Pi} (\text{val}(B) - \text{val}(S)) = \texttt{SW}_\Pi\end{aligned}$$

On the other hand, if we take a matching Π^* that maximizes SW, and plug x^{Π^*} into the ϵ-primal, we have

$$\sum_{(B,S,\in)E} (w_{B,S} - \epsilon)x_{B,S} = \sum_{(B,S,\in)E} w_{B,S}x_{B,S} - \sum_{(B,S,\in)E} \epsilon x_{B,S} \geq \texttt{SW}_{\Pi^*} - n\epsilon$$

The last inequality comes from the fact that n is an upper bound on the number of possible pairs (i.e., number of possible 1's in $x_{B,S}$) for any matching. By the weak duality, any value of ϵ-primal is less than or equal to any value of ϵ-dual, thus $\texttt{SW}_\Pi \geq \texttt{SW}_{\Pi^*} - n\epsilon$.

We now proceed to prove the condition for an ϵ-stable state to maximize SW. Fix a matching Π^* that maximizes SW. Construct graph $G'(V', E')$ with vertices $V' = \mathcal{B} \cup \mathcal{S}$ and edges

$$E' = \{(B, S) \mid (B, S) \in \Pi \vee (B, S) \in \Pi^*\}$$

As any player can be matched with at most one other player in each matching, the degree of each node in G' is at most two. Consequently, the connected components of G' could only be cycles or paths. Note that such cycles and paths are formed by the different pairs of the two matchings. We now prove that for any of those cycles or paths, the local SW of the two matchings are the same.

For any cycle $B_0, S_0, B_1, S_1, \ldots, B_k, S_k, B_0$, pair (B_i, S_i) belongs to one matching while pair (B_{i+1}, S_i) belongs to the other one. If we only consider these players, every buyer gets an item and every seller sells the item, thus the SW of both matchings are the same.

For any path $Z_0, Z_1, Z_2, Z_3, \ldots, Z_k$, wlog, we can assume $\text{val}(Z_0) \geq \text{val}(Z_k)$. If Z_0 is a seller, add his dummy buyer to the left of the path. If Z_k is a buyer, add his dummy seller to the right of the path. Therefore, the path starts with a buyer and ends with a seller. We can denote the path as $B_0, S_0, B_1, S_1, \ldots, B_k, S_k$.

For the same reason as cycle case, the players in the middle contributes same amount of SW to both matchings, thus the difference of SW is $\text{val}(B_0) - \text{val}(S_k)$. Since Π^* is a matching that maximizes SW, it must be the case that $(B_i, S_i) \in \Pi^*$ and $(B_{i+1}, S_i) \in \Pi$.

If the difference of SW is 0, then we are done. Suppose not, then $\text{val}(B_0) - \text{val}(S_k) \geq \delta$. By properties (a_1) and (a_3) of ϵ-stable states,

$$P(B_{i+1}) = P(S_i) \geq P(B_i) - \epsilon \Rightarrow P(B_0) \leq P(B_k) + k\epsilon$$

We now have

$$\begin{aligned} \text{val}(B_0) - \text{val}(S_k) &= P(B_0) - P(S_k) && (2.3) \\ &\leq P(B_k) + k\epsilon - P(S_k) \leq (k+1)\epsilon \leq n\epsilon < \delta && (2.4) \end{aligned}$$

where (3) is because both B_0 and S_k are matched in Π^* but not in Π, implying that their submitted prices are equal to their valuation.

Thus on one side we have $val(B_0) - val(S_k) \geq \delta$, and from the other side by inequality (2.4) above we have $val(B_0) - val(S_k) < \delta$, a contradiction. It concludes that all such cycles and paths generate the same SW for both matchings and thus Π also maximizes SW. $\square$

Note that [40] also shows that a ϵ-stable state realizes an ϵ-approximate SW. However, the bound on ϵ given in Theorem 2.2 is new. In [23], using ϵ-complementary slackness, Bertsekas shows that for integer valuations, any ϵ-stable state achieves maximum SW if $\epsilon < 1/n$. Therefore, for fractional valuations, by scaling valuations with a suitably large factor L, one can make the valuations integers, and deduce that $\epsilon < 1/(nL)$ suffices for

achieving maximum SW. Note that L is at least $1/\delta$ but can possibly be much larger.

We should point out that the bound $\epsilon < \delta/n$ is not an immediate consequence of the fact that any matching in an ϵ-stable state is an ϵ-approximate SW, by arguing that the smallest non-zero difference in SW of two matchings is at least δ. Consider a market whose trading graph is a complete bipartite graph, with four players, where $\text{val}(B_1) = 0.1$, $\text{val}(S_1) = 0.05$ $\text{val}(B_2) = 0.2001$, $\text{val}(S_2) = 0.15$. The difference of valuation price between any two players is lower bounded by 0.05 ($\delta = 0.05$) but B_1, S_1 yields a SW of 0.05 and B_2, S_2 yields a SW of 0.0501 and the difference in SW could be made arbitrarily small.

Finally, it is worth mentioning that the fact that ϵ-stable state gives ϵ-approximate SW does have a corollary as follows, which is a weaker result compared to Theorem 2.2: If for any $(B, S, \in)E$, $\text{val}(B) - \text{val}(S)$ is an integer multiple of δ, then for any $0 \leq \epsilon < \delta/n$, an ϵ-stable state always maximizes SW.

2.5. Convergence to a Stable State

We establish our main results in this section. We will start by describing a mechanism in the spirit of DOA, and show that for any *well-behaved* stable state, there is a sequence of agent moves that leads to this state. When the trading graph is a complete bipartite graph, i.e, the case of the DOA expriment, we show that convergence to a stable state occurs in number of steps that is polynomially bounded in the number agents. However, convergence to a stable state is not guaranteed when the trading graph is an incomplete bipartite graph. We propose a natural randomized extension of our mechanism, and show that with high probability, the market will converge to a stable state in number of steps that is polynomially bounded in the number of agents.

2.5.1. The Main Mechanism

To describe our mechanism, we need the notion of an ϵ-*interested* player.

Definition 2.9 (ϵ-Interested Player)**.** *For a market at state* $\mathbb{S}(P, \Pi)$ *with any parameter*

$\epsilon > 0$, *a seller* S *is said to be* ϵ-interested *in his neighbor* B *iff either* (a) $P(B) \geq P(S)$ *and* $S \notin \Pi$, *or* (b) $P(B) - P(S) \geq \epsilon$ *and* $S \in \Pi$. *The set of buyers interested in a seller* S *is defined analogously.*

When the parameter ϵ is clear from the context, we will simply refer to an ϵ-*interested* player as an interested player.

Mechanism 1. *(with input parameter* $\epsilon > 0$*)*

- ***Activity Rule:*** *Among the unmatched buyers, any buyer that neither submits a new higher bid nor has a seller that is interested in him, is labeled as inactive. All other unmatched buyers are labeled as active. An active (inactive) seller is defined analogously. An inactive player changes his status iff some player on the other side matches with him.*[3]

- ***Minimum Increment:*** *Each submitted price must be an integer multiple of* ϵ. [4]

- ***Recognition:*** *Among all active players, an arbitrary one is recognized.*

- ***Matching:*** *After a buyer* B *is recognized,* B *will choose an interested seller to match with if one exists. If the offer of the seller is lower than the bid* b*, it is immediately raised to* b*. The seller action is defined analogously.*

- ***Tie Breaking:*** *When choosing a player on the other side to match to, an unmatched player is given priority (the* unmatched first *rule).*

In each iteration, players are partitioned into two sets based on whether they are matched or not. The unmatched players are further partitioned into active players and inactive players. The only players with a myopic incentive to revise their submissions are those that are not matched.

Observe that since a buyer will never submit a bid higher than his valuation, and a seller

[3]This is common for eliminating no trade equilibria.
[4]This is part of many experimental implementations of the DOA.

will never make an offer below his own opportunity cost, by submitting only prices that are integer multiples of ϵ, an agent might not be able to submit his true valuation. However, since an agent can always submit a price at most ϵ away from the true valuation, if we pretend that the 'close to valuation' prices are true valuations, the maximum SW will decrease by at most $n\epsilon$. By picking $\epsilon' = \epsilon/2$, if the market converges to an ϵ'-stable state, we still guarantee that the SW of the final state is at most $n\epsilon$ away from the maximum SW.

When a buyer B chooses to increase his current bid: if s denotes the lowest offer in the neighborhood of B, and s' denotes the lowest offer of any unmatched seller in the neighborhood of B, then the new bid of B can be at most $\min\{s+\epsilon, s'\}$. We refer to this as the *increment* rule. This may be viewed as a consequence of rationality – there is no incentive for a buyer to bid above the price needed to make a deal with some seller. A similar rule applies to sellers. With a slight abuse of the terminology, we call either rules increment rule. Notice, a player indifferent between submitting a new price and keeping his price unchanged will be assumed to break ties in favor of activity.

Note that the role of the auctioneer in Mechanism (1) is restricted to recognize agent actions, but never select actions for agents. In fact, the existence of an auctioneer is not even necessary for the mechanism to work. Minimum increment can be interpreted as setting the currency of the market to be ϵ. Arbitrary recognition can be achieved by a first come, first served principle. Activity rule and matching are both designed to ensure that players will keep making actions (submitting a new price or forming a valid match) if one exists.

We first prove some properties of Mechanism (1).

Claim 2.5.1. *For any market, if we use Mechanism (1) with any input parameter $\epsilon > 0$ and start from any state that satisfies properties (a_1) and (a_3) of ϵ-stable states, any state reached satisfies properties (a_1) and (a_3) of ϵ-stable states.*

By the increment rule and matching rule respectively, the reached state satisfies properties

(a_1) and (a_3) of ϵ-stable states.

If a state $\mathcal{S}(P, \Pi)$ satisfies $\forall (B, S, \in) E, P(B) \leq P(S)$, then we call it a *valid starting state*. Note that a valid starting state satisfies properties (a_1) and (a_3) of ϵ-stable states (a valid Π matches a buyer to a seller only if the bid price of the buyer is at least the offer price of the seller). In the following, we only consider markets that begin with a valid starting state, and hence a matched player will never have a move to increase his utility by more than ϵ.

Claim 2.5.2. *For any market, if we use Mechanism (1) and begin with a valid starting state, then any final state of the market is ϵ-stable.*

Claim 2.5.1 ensures that the final state satisfies properties (a_1) and (a_3) of ϵ-stable states, and property (a_2) of ϵ-stable states holds because an unmatched buyer will always submit a new higher bid to avoid being inactive, unless he reaches his valuation. Same for the unmatched sellers.

Note that by Theorem 2.2, if a market converges to an ϵ-stable state, it always realizes ϵ-approximate SW.

Definition 2.10 (Well-behaved)**.** *A stable state $\mathcal{S}(P, \Pi)$, is* well-behaved *iff (a_1) for any $(B, S, \in) E$, if $B \notin \Pi$ and $S \notin \Pi$, then $P(B) < P(S)$. An ϵ-stable state $\mathcal{S}(P, \Pi)$, is* well-behaved *iff not only property (a_1) is satisfied but also (a_2) for any $(B, S, \in) E$, if either $B \notin \Pi$ or $S \notin \Pi$, then $P(B) \leq P(S)$.*

Note that the states ruled out by properties (a_1) and (a_2) of well-behaved states are the corner cases where a buyer-seller pair having the same valuation (thus having no contribution to SW) are not chosen in the matching, or players who can obtain utility at most ϵ stop attempting to match with others.

Theorem 2.3. *For any $\epsilon > 0$, if we use Mechanism (1), and start from the zero-information state, any well-behaved ϵ-stable state can be reached via a sequence of at most n moves. Hence, any well-behaved stable state is also reachable.*

Proof. Given an ϵ-stable state $\mathcal{S}(P, \Pi)$, sort all pairs in Π in decreasing order of prices (arbitrarily break the ties), and denote the ordering as O. We propose a two-stage procedure: first stage handles the players in the matching and second stage deals with the remainders. Note that we only need to justify that the increment rule and unmatched first rule hold for every action.

In stage one, choose pairs of players following the order defined by O. For each pair (B, S), let the buyer submit $P(B)$, and then, let the seller submit $P(S) = P(B)$ and match with B. When B submits $P(B)$, no seller is submitting a price lower then $P(B)$, hence the increment rule is satisfied. The unmatched sellers are submitting 1, and hence either no one is interested in B or all of them are interested in B (if $P(B) = 1$, i.e., $P(B)$ is no less than the seller prices). In the later case, B can directly match with S.

For S, assume the highest bid he can see in his neighborhood is $P(B')$ submitted by buyer B'. By property (a_1) of ϵ-stable states, $P(B') \leq P(S) + \epsilon = P(B) + \epsilon$. Among the unmatched neighbors of S, B is the one submitting the highest price, and $P(S) = P(B) \geq \max\{P(B') - \epsilon, P(B)\}$, the increment rule is satisfied. Since S matches with unmatched buyer B, the unmatched first rule is also satisfied.

In stage two, choose the unmatched players with an arbitrary order and let them submit their valuations. For any unmatched buyer B, by property (a_2) of well-behaved states, $P(S) \geq P(B)$ for any seller S visible to B, hence the increment rule is satisfied. In addition, for any unmatched seller S, by property (a_1) of well-behaved states, $P(B) < P(S)$, thus B cannot match with S. By analogy, any unmatched seller will also make a valid move and remain unmatched.

Thus, after exactly n steps the two stages end, and the market is in state $\mathcal{S}(P, \Pi)$. □

2.5.2. Complete Bipartite Graphs

We now prove that market with complete bipartite trading graph will always converge when using Mechanism (1).

Theorem 2.4. *For a market whose trading graph is a complete bipartite graph, if we use Mechanism (1) with any input parameter $\epsilon > 0$, and begin with any valid starting state, then the market will converge to an ϵ-stable state after at most n^3/ϵ steps.*

We need the following lemma to prove Theorem 2.4.

Lemma 2.5.3. *For a market $G(\mathcal{B}, \mathcal{S}, E, val)$ whose trading graph is a complete bipartite graph, if we use Mechanism (1) with any input parameter $\epsilon > 0$, then at any state $\mathcal{S}(P, \Pi)$ reached from a valid starting state, for any $(B, S, \in)E$, if $P(B) > P(S)$, then both B and S are matched.*

Proof. Assume by contradiction that there exists some $(B, S, \in)E$ with $P(B) > P(S)$ and, wlog, B being unmatched. Since in the starting state, $P(B) \leq P(S)$, let t be the first time that this happens. Therefore, at time $t-1$, either $P(B) \leq P(S)$ or B is matched. Note that since the prices are integer multiples of ϵ, a state with $P(B) > P(S)$ implies $P(B) - P(S) \geq \epsilon$. On the other hand, since property (a_1) of ϵ-stable states always holds, $P(B) - P(S) \leq \epsilon$. Thus $P(B) = P(S) + \epsilon$ at time t.

If $P(B) \leq P(S)$ at time $t-1$, $P(B) > P(S)$ can only be a consequence of either B or S being recognized. If B is recognized and submits a bid of $P(S) + \epsilon$, since S is interested in B, by the matching rule, B will be matched. If S is recognized and submits an offer of $P(B) - \epsilon$, by the increment rule, B must be matched (otherwise S would not submit an offer lower than $P(B)$), a contradiction.

Assume that B was matched to some seller S' at time $t-1$. The only valid action at time $t-1$ that can make B unmatched is if some buyer B' overbids B and match with seller S'. If $S = S'$, then after the move, $P(B) < P(S)$, a contradiction. If $S \neq S'$, then this

move will not change the bid of B or offer price of S, and hence, $P(B) = P(S) + \epsilon$ in time $t-1$. Since the trading graph is a complete bipartite graph, S is a neighbor of B'. By the increment rule, B' can only submit a price at most equal to $P(S) + \epsilon = P(B)$, thus B' is unable to overbid B, a contradiction. □

Definition 2.11 (γ-feasible)**.** *A market state $\mathcal{S}(P, \Pi)$ is said to be γ-feasible iff there are exactly γ matches in Π.*

Proof of Theorem 2.4. Assume at any time t, the state of the market is γ^t-feasible. Define the following potential function

$$\Phi_P = \sum_{S_i \in \mathcal{S}} P(S_i) + \sum_{B_i \in \mathcal{B}} (1 - P(B_i))$$

Note that the value of Φ_P is always an integer multiple of ϵ. We will first show that γ^t forms a non-decreasing sequence over time, and then argue that, for any γ, the market can stay in a γ-feasible state for a bounded number of steps. Specifically, we will show that, if γ does not change, Φ_P is a non-increasing function and can stay unchanged for at most γ steps. Since the maximum value of Φ_P is bounded by n, it follows that after at most $(\gamma n)/\epsilon$ steps, the market moves from a γ-feasible state to a $(\gamma+1)$-feasible state (or converges).

We argue that γ^t forms a non-decreasing sequence over time. Since any recognized player is unmatched, if the action of an unmatched player Z results in a change in the matching, Z either matches with another unmatched player, or matches to a player that was already matched. In either case, the total number of matched pairs does not decrease.

Furthermore, we prove if γ does not change, then Φ_P is non-increasing. Moreover, the number of successive steps that Φ_P stay unchanged is at most γ.

To see that Φ_P is non-increasing, first note that Φ_P can increase only when either a buyer decreases his bid or a seller increases his offer. Assume an unmatched buyer B is recognized (seller case is analogous), and the price function before his move is P. To increase Φ_P,

since B can only increase his bid, he must increase an offer by overbidding and matching with a seller S, resulting in the two of them submitting the same price b. The buyer bid increases by $b - P(B)$ and the seller offer increases by $b - P(S)$. Since B is unmatched, by Lemma 2.5.3, $P(B) \leq P(S)$, and hence Φ_P will not increase.

We now bound the maximum number of steps for which Φ_P could remain unchanged. A move from a buyer B that does not change Φ_P occurs only when B overbids a matched seller S, where the bid and the offer are equal both before and after the move. We call this a *no-change* buyer move. By analogy, a no-change seller move can be defined.

In the remainder of the proof, we first argue that a no-change buyer move can never be followed by a no-change seller move, and vice versa. After that, we prove the upper bound on the number of consecutive no-change moves to show that Φ_P will eventually decrease (by at least ϵ).

Assume at time t_1, a buyer B_{t_1} made a no-change move and matched with a seller S_{t_1}, who was originally paired with the buyer B'_{t_1}.[5] We prove that no seller can make a no-change move at time $t_1 + 1$. The case that a seller makes a no-change move first can be proved analogously. Suppose at time $t_1 + 1$, a seller S_{t_1+1} is recognized and decreases his offer by ϵ. Since B_{t_1} made a no-change move, we have

$$P^{t_1}(B'_{t_1}) = P^{t_1}(B_{t_1}) \tag{2.5}$$

Denote the lowest seller offer (highest buyer bid) at any time t by s^t (b^t). By Lemma 2.5.3, $P^{t_1}(B_{t_1}) \leq P^{t_1}(S)$ for any seller S, hence $P^{t_1}(B_{t_1}) \leq s^{t_1}$. Moreover, since $P^{t_1}(B_{t_1}) = P^{t_1}(S_{t_1}) \geq s^{t_1}$, we have

$$P^{t_1}(B_{t_1}) = s^{t_1} \tag{2.6}$$

In other words, a buyer can make a no-change move, only if his bid is equal to the lowest

[5] An action at time time t will take effect at the time $t+1$, and P^t is the price function before any action is made at time t.

offer. Similarly, if S_{t_1+1} can make a no-change move at time t_1+1, his offer is equal to the highest bid. Since the highest bid at time t_1 (b^{t_1}) is at most $s^{t_1}+\epsilon$ (property (a_1) of ϵ-stable states), after B_{t_1} submits a bid of $s^{t_1}+\epsilon$, he will be submitting the highest bid at time t_1+1. Hence

$$P^{t_1+1}(S_{t_1+1}) = b^{t+1} = P^{t_1+1}(B_{t_1}) = P^{t_1+1}(B'_{t_1}) + \epsilon \tag{2.7}$$

Therefore, at time t_1+1, after S_{t_1+1} decreases his offer by ϵ, the unmatched buyer B'_{t_1} is interested in S_{t_1+1}. By the unmatched first rule, S_{t_1+1} will match with an unmatched player, hence this cannot be a no-change move.

This proves that a no-change seller move can never occur after a no-change buyer move and vice versa. We now prove the upper bound on the number of consecutive no-change buyer moves.

For any sequence of consecutive no-change buyer moves, if there exists a time t_2 such that $s^{t_2} > s^{t_2-1}$, for any unmatched buyer B at time t_2, $P^{t_2}(B) \leq s^{t_2-1} < s^{t_2}$. By Equation (2.6), no buyer can make any more no-change move. Moreover, since any no-change buyer move will increase the submission of a matched seller who is submitting the lowest offer, after at most γ steps, the lowest offer must increase, implying that the length of the sequence is at most γ.

To conclude, the total number of steps that the market could stay in γ-feasible states is bounded by $(n/\epsilon)\gamma$. As $\gamma \leq n$, the total number of steps before market converges is at most n^3/ϵ. □

2.5.3. General Bipartite Graphs

In this section, we study the convergence of markets with an arbitrary bipartite trading graph. Although by Theorem 2.3, using Mechanism (1), the market can reach any well-behaved ϵ-stable state, when the trading graph of a market can be an arbitrary bipartite

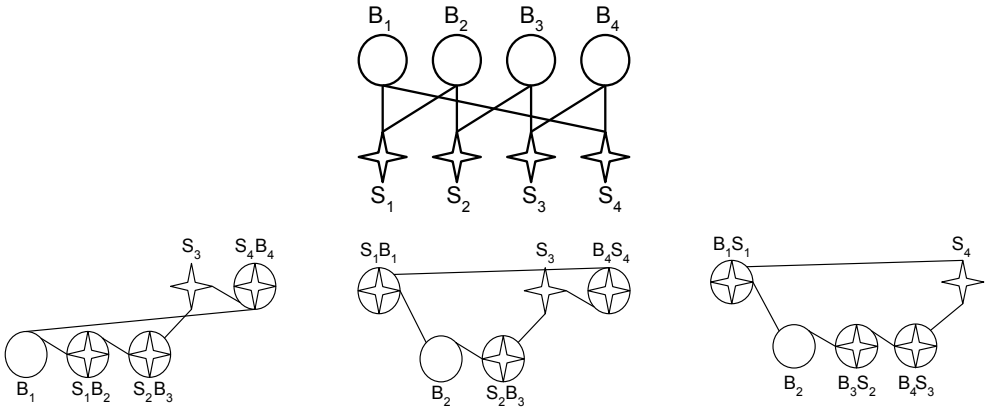


Figure 1: Unstable market with general trading graph and Mechanism (1)

graph, there is no guarantee that Mechanism (1) will actually converge.

Claim 2.5.4. *In a market whose trading graph is an arbitrary bipartite graph, Mechanism (1) may never converge.*

Consider the market shown in Figure 1. In this market, there are four buyers (B_1 to B_4) all with valuation 1 and four sellers (S_1 to S_4) all with opportunity cost 0. Moreover, the trading graph is a cycle of length 8, as illustrated by the first graph in Figure 1. Assume at some time t, the market enters the state illustrated by the second graph, where B_1, B_2, S_1, B_3, S_2 are submitting 5ϵ, S_3, B_4, S_4 are submitting 6ϵ, and pairs (B_2, S_1), (B_3, S_2) and (B_4, S_4) are matched.

At time $t+1$, since B_1 is unmatched, he can be recognized and submit 6ϵ. S_1 is the only interested seller, hence B_1, S_1 will match and the offer of S_1 increases to 6ϵ, which leads to the state shown in the third graph. Similarly, at time $t+2$, since S_3 is unmatched, he can be recognized and submit 5ϵ. B_4 is the only interested buyer, hence B_4, S_3 will match and bid of B_4 increases to 6ϵ, which leads to the state shown in the fourth graph.

Notice that the states at time t and $t+2$ are isomorphic. By shifting the indices and repeating above two steps, the market will never converge.

Observe that the cycle described in Claim 2.5.4 is caused by an adversarial coordination

between the actions of various agents. To break this pathological coordination, we introduce Mechanism (2) which is a natural extension of Mechanism (1) that uses randomization. We first define this mechanism, and then prove that on any trading graph, with high probability, the mechanism leads to convergence in a number of steps that is polynomially bounded in the number of agents.

Mechanism 2. *(with input parameter $\epsilon > 0$)*

- ***Activity Rule:*** *Among the unmatched buyers, any buyer that neither submits a new higher bid nor has a seller that is interested in him, is labeled as inactive. All other unmatched buyers are labeled as active. An active (inactive) seller is defined analogously. An inactive player changes his status iff some player on the other side matches with him.*

- ***Minimum Increment:*** *Each submitted price must be an integer multiple of ϵ.*

- ***Bounded Increment Rule:*** *In each step, a player is only allowed to change his price by ϵ.*

- ***Recognition:*** *Among all players who are active, one is recognized uniformly at random.*

- ***Matching:*** *After a player, say a buyer B, is recognized, if B does not submit a new price, then B will match to an interested seller if one exists. If the offer of the seller is lower than the bid b, it is immediately raised to b. The seller action is defined analogously.*

- ***Tie Breaking:*** *When choosing a player on the other side to match to, an unmatched player is given priority (unmatched first rule).*

Notice that we ask players to move cautiously through the bounded increment rule. Players can either change the price by ϵ or match with an interested seller, and always favor being active. Note that, any move in Mechanism (1) can be simulated by at most $(1/\epsilon + 1)$ moves

in Mechanism (2) ($1/\epsilon$ for submitting new price and 1 for forming a match). The following is an immediate consequence of results shown in Section 2.5.1.

Corollary 2.5. *For any market, if we use Mechanism (2), (i) starting from the zero-information state, any well-behaved ϵ-stable state can be reached in $n(1/\epsilon + 1)$ steps, and (ii) beginning with a valid starting state, properties (a_1) and (a_3) of ϵ-stable states always hold, and the final state is ϵ-stable.*

We are now ready to prove our second main result, namely, for any trading graph, with high probability, Mechanism (2) converges to a ϵ-stable state in a number of steps that is polynomially bounded in the number of agents. We will utilize the following standard fact about random walk on a line (see [90], for instance).

Claim 2.5.5. *Consider a random walk on $\{0, 1, 2, ..., N\}$ such that for any $i \in [1, N]$, the random walk transition from state i to state $(i-1)$ happens with probability α, and for any $i \in [0, N-1]$, the random walk transition from state i to state $(i+1)$ happens with probability β, for some $\alpha + \beta = 1$. Then starting from any $i \in [0, N]$, with probability at least $1/2$, the random walk either reaches the state 0 or the state N, after $O(N^2)$ steps.*

Theorem 2.6. *For any market $G(\mathcal{B}, \mathcal{S}, E, val)$, if we use Mechanism (2) with any input parameter $\epsilon > 0$, and begin with a valid starting state, the market will converge to an ϵ-stable state after at most $O((n^3/\epsilon^2) \log n)$ steps with high probability.*

Proof. Let $u^t_{\mathcal{B}}$ and $u^t_{\mathcal{S}}$ denote the number of active buyers and sellers at time t, respectively, and let $u^t = u^t_{\mathcal{B}} + u^t_{\mathcal{S}}$. We will first show that $u^t_{\mathcal{B}}$ and $u^t_{\mathcal{S}}$ are both non-increasing functions of time and then argue that for any u, with high probability the market will remain in a state with u active players for a number of steps that is polynomially bounded in the number of players.

We first prove $u^t_{\mathcal{B}}$ and $u^t_{\mathcal{S}}$ are non-increasing. Note that the only move that can make a new player active is one where a player, say a buyer B, matches to a currently matched seller S. Let B' be the buyer that is currently matched to S. Then at time $t+1$, the buyer B moves out of the set of active players, while the buyer B' possibly joins the set of active players.

Thus the number of active players remains unchanged. A similar argument applies to case when a seller is recognized and matches to a currently matched buyer.

In the remainder of the proof, we first show that if there exists an adjacent buyer-seller pair such that both players are unmatched and the buyer bid is not below the seller offer (we call such a pair to be an *active pair*), then after $O(n \log n)$ steps, with probability $1 - O(1/n^2)$, u^t will decrease. Next, in the absence of active pairs, we argue that either u^t decreases or an active pair appears in the market after $O((n/\epsilon)^2 \log n)$ steps, with probability $1 - O(1/n^2)$. Note that if an active pair appears, by the same argument, after $O(n \log n)$ more steps, u^t will decrease with high probability. Since $u^t \leq n$, we can conclude that the market converges in $O((n^3/\epsilon^2) \log n)$ steps with high probability.

We first prove that, the existence of an active pair will lead to decrement of u^t. For any active pair (B, S). By the unmatched first rule, recognizing either B or S will increase the number of matches. Recognizing any other player who makes a move to match with B or S, will also increase the number of matches (note that only unmatched players will be recognized). Both cases decrease u^t by 2. In other words, as long as u^t does not decrease, (B, S) will remain to be an active pair. Assume at time t_1, there is an active pair (B, S). Let Y_t be the random variable which is 1 iff B or S is recognized at time t. Hence, for any $t \geq t_1$ where $u^t = u^{t_1}$,

$$Pr(Y_t = 1) = \frac{2}{u^{t_1}} \geq \frac{2}{n}$$

It follows that after n steps from t_1 the probability that none of B or S is chosen is less than $1/2$. Therefore, after $2n \log n$ steps, with probability $1 - (1/n)^2$, either one of B and S has been recognized or u^t has already decreased. In either case, u^t decreases.

Next, in the absence of active pairs, we prove that after a bounded number of steps, either u^t decreases or an active pair appears. Consider the following potential function

$$\Phi = \sum_{B \in \mathcal{B}} P(B) + \sum_{S \in \mathcal{S}} P(S)$$

If there is no active pairs, by the design of the Mechanism (2), when recognized, any buyer will increase Φ by ϵ and any seller will decrease Φ by ϵ. Thus, at any time t with no active pairs, the probability that Φ increases by ϵ is $P_{\mathcal{B}}^t = u_{\mathcal{B}}^t/u^t$, and the probability that Φ decreases by ϵ is $P_{\mathcal{S}}^t = u_{\mathcal{S}}^t/u^t$ (note that $P_{\mathcal{B}}^t$ or $P_{\mathcal{S}}^t$ might be 0).

In the following, we will use Claim 2.5.5 to prove that as long as u^t does not change and no active pair appears, after a bounded number of steps, with high probability, Φ will have reached its upper or lower bound. If Φ reaches its upper bound then all buyers must be submitting their true valuations and all sellers must be submitting 1. Thus every unmatched buyer is inactive and u^t must have decreased. A similar situation also happens when Φ reaches its lower bound.

To use Claim 2.5.5, see that if u^t does not change and no active pair appears, $P_{\mathcal{B}}^t$ and $P_{\mathcal{S}}^t$ will also remain unchanged. During this time period, we can denote the probability of Φ increases by ϵ as $P_{\mathcal{B}}$ and the probability of Φ decreases by ϵ as $P_{\mathcal{S}}$. Let $\alpha = P_{\mathcal{S}}$, $\beta = P_{\mathcal{B}}$, and nodes be $\{0, \epsilon, 2\epsilon, \ldots, n\}$ (hence $N = n/\epsilon$). Thus this is a random walk, and by Claim 2.5.5, after $O((n/\epsilon)^2)$ steps, the probability of Φ reaches its upper bound n or lower bound 0 is at least 1/2. Therefore, after $O((n/\epsilon)^2 \log n)$ steps, with probability at least $1 - O(1/n^2)$, Φ reaches 0 or n.

To conclude, after $O((n/\epsilon)^2 \log n)$ steps, u^t will decrease with probability at least $1 - O(1/n^2)$. As $u^t \leq n$, by union bound, the market will converge after $O((n^3/\epsilon^2) \log n)$ steps with probability $1 - O(1/n)$.

□

2.6. Conclusions and Future Work

In this chapter, we resolved the second part of Friedman's conjecture by designing a mechanism which simulates the DOA and proving that this mechanism always converges to a Walrasian equilibrium in polynomially many steps. Our mechanism captures four key

properties of the DOA: agents on either side can make actions; agents only have limited information; agents can choose *any* better response (as opposed to the best response); and the submissions are recognized in an arbitrary order. An important aspect of our result is that, unlike previous models, *every* Walrasian equilibrium is reachable by some sequence of better responses.

For markets where only a restricted set of buyer-seller pairs are able to trade, we show that the DOA may never converge. However, if submissions are recognized randomly, and players only change their bids and offers by a small fixed amount, convergence is guaranteed. It is unclear that the latter condition is inherently necessary, and it would be an interesting future work to obtain a convergence result for a relaxed notion of bid and offer changes where players can make possibly large adjustments as long as they are consistent with the increment rule. Another interesting direction of future work is to understand if similar convergence results can established for the more general trading model where, instead of having the notion of buyers and sellers, any agents can trade with each other.

CHAPTER 3 : Stochastic Matching

Our work on the double oral auction for the simple buyer-seller trading game, discussed in the previous chapter, mainly focused on establishing convergence time. In this chapter, we will switch to another simple trading game, i.e., the stochastic matching problem, where our focus will (temporarily) switch to data summarization[1]. We will first formally define the stochastic matching problem and two types of algorithms for solving the stochastic matching problem, namely *adaptive* algorithms and *non-adaptive* algorithms. Then, we design an adaptive and a non-adaptive algorithms respectively for the stochastic matching problem which achieves the same approximation ratio as the previous best bound [25] with a much better degree bound. We will start by defining and briefly motivation the stochastic matching problem.

3.1. Background

The *stochastic matching* problem concerns the problem of finding a maximum matching in presence of *uncertainty* in the input graph. Specifically, we are given an undirected graph $G(V, E)$ where each edge $e \in E$ is *realized* with some constant probability $p > 0$ and the goal is to find a maximum matching in the realized graph. To find a large matching, an algorithm is allowed to *query* edges in E to determine whether or not they are realized.

Obviously, there is a trivial solution for the stochastic matching problem: simply query all edges in E to see which edges are realized and compute a maximum matching over the realized graph. However, in many applications, determining whether or not an edge is realized could be both costly and time consuming (we will elaborate more on this later in this section). Consequently, to minimize cost, it is preferable that an algorithm queries as few edges as possible, and to minimize the time consumed in the query process, an algorithm should have only a few rounds of *adaptivity* whenever possible (or minimize the *degree of adaptivity*), meaning that it is preferable if the decision of "which edges to query next"

[1]The full paper of this work can be found in [13].

does not depend on the outcome of previous queries since in this case, many edges can be queried in parallel. A more formal definition of this model is presented in Section 3.2.

3.1.1. Kidney Exchange

A canonical and arguably the most important application of the stochastic matching problem appears in *kidney exchange*. Typically, organ donation comes from deceased donors since many organs cannot be harvested from a living donor without jeopardizing the health of the donor. Fortunately, kidney is an exception, and a healthy living donor is able to donate one of his/her two kidneys without facing major life threatening consequences.

The possibility of having living donors triggers the idea of kidney exchange: often patients have a family member who is willing to donate his/her kidney, but this kidney might not be a suitable match for the patient due to reasons like incompatible blood-type etc. To solve this problem, a kidney exchange is performed in which patients *swap* their incompatible donors to each get a compatible donor.

This setting can be modeled as a maximum matching problem as follows. Create a graph $G(V, E)$ where each patient-donor pair is a vertex and there is an edge between two vertices iff the two patient-donor pairs can perform a kidney exchange. Consequently, a maximum matching in this graph identifies the maximum number of patient-donor pairs that are able to perform an exchange.

To construct such a graph for kidney exchange, typically we only have access to the medical record of the patients and the donors, which contains information like blood type, tissue type, etc. This information can be used to rule out the patient-donor pairs where donation is impossible (e.g., different blood types), but does not provide a conclusive answer for whether or not a donation is indeed feasible. In order to be (more) certain that a donor can donate to a patient, more accurate tests must be performed before the transplant, that includes crossmatching, antibody screen, etc[2], which are both costly and time consuming.

[2]American Transplant Foundation, `http://www.americantransplantfoundation.org/`.

The stochastic matching problem captures the essence of the need of extra tests for kidney exchange: an algorithm selects a set of patient-donor pairs to perform the extra costly and time consuming tests (i.e., query the edges), while making sure that w.h.p. there is a large matching among the pairs that pass the extra tests (i.e., in the realized graph). The objective of querying as few edges as possible captures the essence of minimizing the total cost and the objective of having small degree of adaptivity captures the essence of minimizing patient's waiting time by performing the extra exams between many patient-donor pairs in parallel.

The kidney exchange problem has been extensively studied in the literature, particularly under stochastic settings (see, e.g., [43, 12, 86, 106, 8, 11, 17, 42, 44]); we refer the interested reader to [25] for a more detailed discussion.

3.2. Our Results and Related Work

We now formally define the model for the stochastic matching problem. For any graph $G(V, E)$, let $\text{opt}(G)$ denote the *maximum matching size* in G. With a slight abuse of notation, we sometimes also use $\text{opt}(X) := \text{opt}(G(V, X))$ for $X \subseteq E$, i.e., X is a set of edges instead of a graph. Throughout, we use n to denote $|V|$.

In the stochastic setting, for the input graph $G(V, E)$, each edge in E is realized *independently* w.p.[3] p. When sampling each edge w.p. p, for any set of edges $X \subseteq E$, we slightly abuse the notation and use X_p to denote both a random variable that corresponds to sampling edges in X w.p. p, as well as a specific realization of the random variable X_p. We call each possible realized graph $G(V, E_p)$ a *realization* of G.

In the *stochastic matching problem*, we are given a graph $G(V, E)$ and our goal is to compute a matching M in $G(V, E_p)$, such that w.h.p. (taken over both the randomness of the algorithm and the randomness of the realization $E_p \subseteq E$), the size of M is close to $\text{opt}(E_p)$. An algorithm is allowed to *query* any edge $e \in E$ to determine whether or not $e \in E_p$, and

[3] Throughout, we use *w.p.* and *w.h.p.* to abbreviate "with probability" and "with high probability", respectively.

we consider the following two classes of algorithms.

- **Non-adaptive algorithms.** A non-adaptive algorithm specifies a subset of edges $Q \subseteq E$, queries all edges in Q in parallel, and outputs a matching among the edges realized in Q.

- **Adaptive algorithms.** An adaptive algorithm proceeds in *rounds* where in each round, based on the edges queried and realized thus far, the algorithm chooses a new set of edges to query in parallel. We say the *degree of adaptivity* of an algorithm is d if the algorithm makes at most d rounds of adaptive queries.

In general, the goal is to design algorithms where the number of per-vertex queries is independent of n. For adaptive algorithms, the degree of adaptivity is further required to be independent of n.

We remark that throughout, we always assume $\text{opt}(G) = \omega(1/p)$ to obtain the desired concentration bounds. This assumption essentially says that the expected matching size in the realized graph is bounded from below by a sufficiently large constant.

3.2.1. Related Work

Prior to our work, the state of the art adaptive and non-adaptive algorithms for stochastic matching are that of [25], which is an adaptive (resp. non-adaptive) algorithm that achieves a $(1-\epsilon)$-approximate (resp. $(\frac{1}{2}-\epsilon)$-approximate) matching *in expectation*, while both the number of per-vertex queries and the degree of adaptivity (for the adaptive algorithm) is $\frac{1}{p^{O(1/\epsilon)}}$. Note that while in these algorithms, the number of per-vertex queries and degree of adaptivity is independent of n, the exponential dependence on $\frac{1}{\epsilon}$, limits the practical appeal of the algorithm. [25] raised an open problem regarding the possibility of avoiding the exponential dependency on $\frac{1}{\epsilon}$ for both the number of per-vertex queries and the degree of adaptivity.

Other variants of the stochastic matching setting have also been studied in the literature.

[26] considered the setting where each vertex can only pick two incident edges to query and the goal is to find the optimal set of edges to query. Another well studied setting is the *query-commit* model, whereby if an algorithm decides to query an edge e, then e must be part of the matching the algorithm outputs in case e is realized [38, 7, 30, 19, 65].

3.2.2. Our Results

We provide algorithms for the stochastic matching problem with exponentially smaller number of queries and degree of adaptivity (for the adaptive algorithm) compared to the best previous bounds of [25]. In particular,

Theorem 3.1 (Informal)**.** *For any $\epsilon > 0$, there exists a poly-time adaptive $(1-\epsilon)$-approximation algorithm for the stochastic matching problem which queries $O\left(\frac{\log(1/\epsilon p)}{\epsilon p}\right)$ edges per vertex and has degree $O\left(\frac{\log(1/\epsilon p)}{\epsilon p}\right)$ of adaptivity.*

The formal statement of Theorem 3.1 is presented in Section 3.5 (as Theorem 3.3).

Theorem 3.2 (Informal)**.** *For any $\epsilon > 0$, there exists a poly-time non-adaptive $(\frac{1}{2} - \epsilon)$-approximation algorithm for the stochastic matching problem which queries $O\left(\frac{\log(1/\epsilon p)}{\epsilon p}\right)$ edges per vertex.*

The formal statement of Theorem 3.2 is presented in Section 3.6 (as Theorem 3.4).

These results provide an affirmative answer to an open question raised by [25] regarding the possibility of avoiding the exponential dependency on $1/\epsilon$ for both the number of per-vertex queries and the degree of adaptivity.

One of the key property of our results is that we provide *instance-wise* approximation guarantees, i.e., for $1 - o(1)$ fraction of the realizations of G, the algorithm outputs a competitive solution. This is a *stronger* guarantee than the *expectation* guarantee provided in [25] which states that expected size of the matching output by the algorithm is competitive with the expected size of the maximum matching among all realizations. We remark that our instance-wise guarantee is of particular interest to the key application of the kidney

exchange problem.

Finally, it is worth mentioning that even when the input graph is a complete graph, one needs to query $\Omega(\log(1/\epsilon)/p)$ edges per each vertex to simply ensure that the number of isolated vertices in the realized graph is at most ϵn. This is due to the fact that if one queries less than $\ln(1/\epsilon)/2p$ edges per vertex, the probability that no edge incident on a vertex is realized is $(1-p)^{\ln(1/\epsilon)/2p} \geq \exp(-2p \cdot \ln(1/\epsilon)/2p) = \epsilon$. On the other hand, it is easy to see that for any constant $p > 0$, any realization of a complete graph has a perfect matching w.h.p. Hence, $\Omega(\log(1/\epsilon)/p)$ is a simple lower bound on the number of per-vertex queries for any $(1-\epsilon)$-approximation algorithm even on complete graphs. Our per-vertex query bounds in Theorem 3.1 and Theorem 3.2 only ask for slightly more than this simple lower bound.

3.2.3. Our Techniques

To explain the high-level idea underlying our algorithms, it will be convenient to focus on the case when G_p has a perfect matching; however, we emphasize that our algorithms do not require this property. The idea behind both of our algorithms is to construct a *matching-cover* of the input graph G and query the edges of the cover in the algorithm. Roughly speaking, a γ-matching-cover of a graph $G(V, E)$ is a collection of matchings of G of size $\gamma \cdot (|V|/2)$ that are essentially edge-disjoint (see Section 3.4.1 for a formal definition). One of the main technical ingredient of our work is a structural result proving that: for any algorithm that outputs a γ-matching cover with $\Theta(1/\epsilon p)$ matchings, w.h.p. the set of realized edges in the cover contains a matching of size $(1-\epsilon) \cdot \gamma \cdot (|V|/2)$. We prove this result through a constructive argument based on the Tutte-Berge formula (see Section 3.3 for more details on Tutte-Berge formula).

Next, we show that there is a simple adaptive (resp. non-adaptive) algorithm that computes a 1-matching-cover (resp. 1/2-matching-cover) with $\Theta(1/\epsilon p)$ matchings, which immediately implies that w.h.p. the algorithm achieves a $(1-\epsilon)$-approximation (resp. $(\frac{1}{2}-\epsilon)$-

approximation).

Finally, to eliminate the assumption that G_p has a perfect matching, we establish a *vertex sparsification lemma* (see Section 3.4.2) which allows us to reduce the number of vertices in any instance G from $|V|$ to $O(\text{opt}(G)/\epsilon)$, while w.h.p. preserving the maximum matching size to within a factor of $(1-\epsilon)$. In the sparsified graph $\mathcal{G}(\mathcal{V},\mathcal{E})$, although we only have $\text{opt}(G_p) = \Omega(|\mathcal{V}|)$ instead of having a perfect matching, we can show that, with some more care in the analysis, the constant gap between $\text{opt}(G_p)$ and $|\mathcal{V}|$ is enough for us to establish the approximation ratios of our algorithms.

Comparison with [25] The adaptive algorithm of [25] can be summarized as follows: maintain a matching M, and at each round find a collection of vertex-disjoint *augmenting paths* of length $O(1/\epsilon)$ in the input graph $G(V,E)$ with respect to M; query the edges of the augmenting paths and augment M if possible. Using the well-known fact that a matching with no augmenting path of length $O(1/\epsilon)$ is a $(1-\epsilon)$-approximate matching, the authors show that the found matching of the algorithm is a $(1-\epsilon)$-approximation in expectation.

In this process, the probability that an augmenting path of length $O(1/\epsilon)$ "survives" the querying process is only $p^{O(1/\epsilon)}$; hence, one needs to repeat the whole process roughly $\frac{1}{p^{O(1/\epsilon)}}$ times, which leads to the same degree of adaptivity and per-vertex queries. The non-adaptive algorithm of [25] is designed based on similar framework of using augmenting paths.

On the other hand, our algorithms exploit the structure of matchings in a global way (using the Tutte-Berge formula) instead of locally searching for short augmenting paths. In particular, through the use of matching covers, we completely eliminate the need of searching for the augmenting paths and hence avoid the exponential dependency on $\frac{1}{\epsilon}$, which is essentially the length of the augmenting paths. It is worth mentioning that however most of these differences only appear in the analysis; the description of our algorithms and algorithms of [25] are both simple and similar (modulo the extra sparsification part of our

algorithm).

Organization: The rest of this chapter is organized as follows. In Section 3.3, we introduce some notation and preliminaries used in our analysis. In Section 3.4, we establish our key lemmas which play an important role in both our adaptive and non-adaptive algorithms. The description and analysis of our adaptive algorithm and our non-adaptive algorithm are presented in Section 3.5 and Section 3.6, respectively. Finally, we introduce in Section 3.7 a barrier for obtaining a better approximation using non-adaptive algorithms, and conclude in Section 3.8 with some future directions.

3.3. Preliminaries

Notation. Throughout for any set of edges $X \subseteq E$, $V(X)$ denotes the set of vertices incident on X. For two integers $a \leq b$, $[a, b]$ denotes the set $\{a, \ldots, b\}$ and $[b] := [1, b]$.

Tutte-Berge formula. In our proofs, we crucially rely on the Tutte-Berge formula which generalizes the *Hall's marriage theorem* for characterizing perfect matchings in bipartite graphs to maximum matchings in general graphs. For any graph $G(V, E)$ and any $U \subseteq V$, $odd(V - U)$ denotes the number of connected components with *odd* number of vertices in $G(V \setminus U, E)$. We have,

Lemma 3.3.1 (Tutte-Berge formula)**.** *The size of a maximum matching in a graph $G(V, E)$ is equal to*

$$\frac{1}{2} \min_{U \subseteq V} \Big(|U| + |V| - odd(V - U) \Big)$$

See, e.g., [84] (Chapter 3) for a proof of this lemma.

Finally, we have the following simple concentration result on the size of a maximum matching in G_p.

Claim 3.3.2. *For any graph $G(V, E)$ with $opt(G) = \omega(1/p)$,*

$$\Pr\left(opt(G_p) \geq p \cdot opt(G)/2\right) = 1 - o(1).$$

Proof. Fix any maximum matching M in G. $\mathrm{E}\left[|M_p|\right] = p \cdot |M| = p \cdot \mathrm{opt}(G)$. Hence, by Chernoff bound, w.p. $1 - o(1)$, $|M_p| \geq \mathrm{E}\left[|M_p|\right]/2 \geq p \cdot \mathrm{opt}(G)/2$. Noting that M_p is also a matching in G_p concludes the proof. □

3.4. Matching Covers and Vertex Sparsification

In this section, we present our main technical results, namely the *matching-cover lemma* and the *vertex sparsification lemma*, which lie in the core of both our adaptive and non-adaptive algorithms. We start by describing the matching-cover lemma. As explained earlier, the matching-cover lemma is already sufficient for establishing the approximation guarantee of the algorithms as long as $\mathrm{opt}(G) = \Omega(n)$. To tackle the case where $\mathrm{opt}(G)$ is much smaller than the number of vertices, we next introduce a simple vertex sparsification lemma that provides a way of reducing the number of vertices in any graph $G(V, E)$ from $|V|$ to $O(\mathrm{opt}(G))$ while preserving the maximum matching size approximately, for *any* realization of $G(V, E_p)$.

3.4.1. Matching-Cover Lemma

We start by defining the following process which takes any graph as an input, and outputs a list of matchings (i.e., a *matching-cover*).

Definition 3.1 (Incremental Matching Selection Process)**.** *We say an algorithm is an* incremental matching selection process (IMSP) *iff for any input graph G, the algorithm selects a sequence of matchings $M_1, M_2, \ldots, M_r$ one by one from G such that for any $i \in [r]$, for any edge e selected in the i-th matching M_i where e also appears in M_j for some $j < i$, the edge e must be realized.*

We refer to the matchings an IMSP outputs as a *matching-cover*, and we say that an IMSP

outputs a γ-*matching-cover* iff for all $i \in [r]$, $|M_i| \geq \frac{\gamma n}{2}$. The following claim states the key property of any IMSP, which will be used in our proofs.

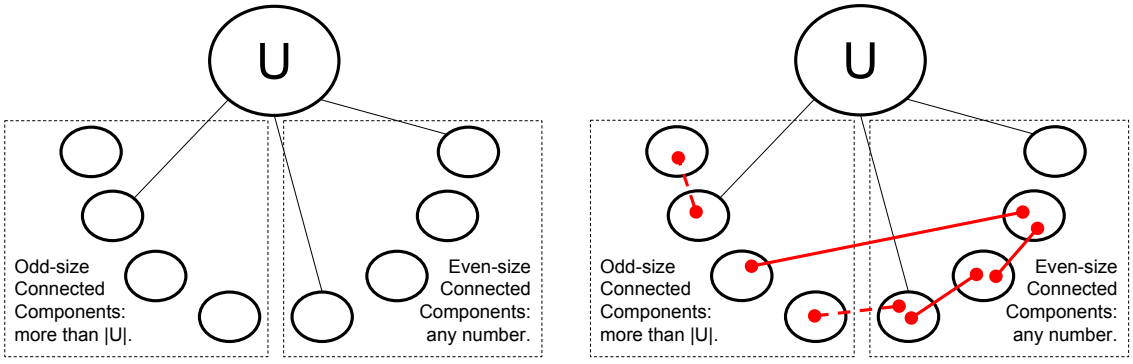
Claim 3.4.1. *For any IMSP A and any graph G, denote by $M_1, M_2, \ldots, M_r$ the matching-cover that A outputs on G; then for any $i \in [r]$, conditioned on any realization of $M_1, M_2, \ldots, M_{i-1}$, and any choice of the matching M_i, each edge $e \in M_i$ is realized w.p. at least p, independent of any other edges in M_i.*

Proof. For the edges $e \in M_i$ that appear in previous matchings, by the definition of IMSP, e is realized w.p. $1 (\geq p)$, which is trivially independent of any other edges in M_i. For the set of edges E' that do not appear in any previous matching, E' is disjoint with $M_1, M_2, \ldots, M_{i-1}$ and the set of realized edges in $M_1, M_2, \ldots, M_{i-1}$ is independent of realization of edges in E'. Therefore, by the definition of the stochastic setting, each edge in E' is realized w.p. p, independent of other edges. □

We are now ready to state the matching-cover lemma, which is the main result of this section.

Lemma 3.4.2 (Matching-Cover Lemma)**.** *For any parameter $0 < \epsilon, p < 1$, any graph G, and any IMSP A, denote by $M_1, \ldots, M_r$ the γ-matching-cover of G that A outputs. If $r \geq \frac{32 \log(2e/\gamma)}{\epsilon p}$ and $\gamma \cdot n = \omega(1)$, then, w.p. $1 - o(1)$, there is a matching of size at least $(1-\epsilon)\frac{\gamma n}{2}$ among the realized edges in the matching-cover.*

We remark that Lemma 3.4.2 holds even for multi-graphs, which is a property required by our algorithms (due to the usage of vertex sparsification). We first provide a high-level summary of the proof. Suppose by contradiction that the output of IMSP A does not contain a large matching; then, by Tutte-Berge formula, there should exist a set of vertices $U \subseteq V$ where removing U from the graph results in many connected components (CC) with odd number of vertices, namely U is an *odd-sets-witness* (see Fig 2.a for an example of an odd-sets-witness). Our strategy is to show that, for any fixed set U, the probability that U ends up being an odd-sets-witness is sufficiently small, and then use a union bound over all



(a) A candidate odd-sets-witness U.

(b) Thick (red) edges are matching M_i. Solid edges are realized while dashed edges are not.

Figure 2: An example of adding a large matching M_i to an odd-sets-witness (the red/thick edges). The number of odd components does not decrease but the total number of connected components indeed decreases.

possible choices of U to argue that w.h.p. no such odd-sets-witness can arise.

To see that w.h.p, each U does not lead to an odd-sets-witness, we again use the Tutte-Berge formula: if the edges realized in the first i matchings leave many odd-size CC's, then the large matching M_{i+1} must eliminate most of them. Note that this is not enough to show that the number of *odd-size CC's* will decrease w.h.p. since it is possible that M_{i+1} eliminates two odd-size CC's by connecting them through a chain of even-size CC's (see the long chain in Fig 2.b for an illustration). The length of the chain could be arbitrarily long and even if one edge on the chain is not realized, the two odd-size CC's will still end up being disconnected. A key observation here is that though the number of odd-size CC's might not decrease (requiring all edges on the chain to be realized), but the total number of CC's will decrease w.h.p. (any realized edge on the chain reduces the number of CC's). Using this fact, we show that after enough number of rounds, the total number of CC's will drop significantly and even if all of them are odd-size CC's, it is not enough for being a odd-sets-witness.

of Lemma 3.4.2. If the edges realized in the matching-cover do not contain a matching of size more than $(1-\epsilon)\frac{\gamma n}{2}$, then, by Lemma 3.3.1, there exists a set of vertices U where the

number of odd-size connected components after removing U, i.e., $odd(V-U)$, satisfies

$$(1-\epsilon)\frac{\gamma n}{2} \geq \frac{1}{2}\Big(|U|+|V|-odd(V-U)\Big)$$

which implies that $odd(V-U) \geq |U| + (1-\gamma+\epsilon\gamma)n$. In this case, we say U leads to an *odd-sets-witness* and denote this event by $\mathcal{E}_U$. Using this fact, we only need to prove the following lemma.

Lemma 3.4.3. *For any $U \subseteq V$, $Pr[\mathcal{E}_U] \leq 2^{-2\gamma n \log(2e/\gamma)}$.*

We first show that Lemma 3.4.3 implies Lemma 3.4.2. We will apply a union bound over all candidate sets U to show that probability that *there exists* some U where $\mathcal{E}_U$ happens is $o(1)$. In order to so, we argue that the number of different choices of U's that need to be considered is at most $2^{\gamma n \log(2e/\gamma)}$. To see this, note that every odd-size set must contain at least one vertex, and there are only n vertices that could be part of the odd-size sets. Thus, we have $n \geq odd(V-U)$. On the other hand, as we just established, $odd(V-U) \geq |U| + (1-\gamma+\epsilon\gamma)n$, and combining the two inequalities, we have

$$|U| \leq n - (1-\gamma+\epsilon\gamma)n \leq \gamma n \tag{3.1}$$

Therefore, it suffices to consider the sets U with cardinality at most γn, and only

$$\binom{n+\gamma n}{\gamma n} \leq \Big(\frac{e(1+\gamma)n}{\gamma n}\Big)^{\gamma n} \leq \Big(\frac{2e}{\gamma}\Big)^{\gamma n} = 2^{\gamma n \log(2e/\gamma)}$$

such choices of U exists. Now, we can apply a union bound over all such choices of U, and by Lemma 3.4.3 w.p. at least $1 - 2^{-\gamma n \log(2e/\gamma)} = 1 - o(1)$ (recall that $\gamma n = \omega(1)$), there is no odd-sets-witness, proving Lemma 3.4.2. It remains to prove Lemma 3.4.3, and as stated in Eq (3.1) we can assume, wlog, that $|U| \leq \gamma n$.

of Lemma 3.4.3. Recall that the goal is to show that w.h.p., U does not lead to an odd-sets-witness (i.e., $\mathcal{E}_U$ does not happen). We first define some notation. For any $i \in [r]$,

$M_p(i)$ denotes the set of edges in M_i that are realized and are not incident on vertices in U, and G_i denotes the graph after realizing the edges in the first i matchings. We use $cc(G_i)$ to denote the number of connected components in G_i and use $\mathcal{E}_i$ to denote the event that the number of odd-size connected components in G_i is at least $|U| + (1 - \gamma + \epsilon\gamma)n$.

Let $\mathcal{Y}_i$ be a random binary variable where $\mathcal{Y}_i = 1$ iff $cc(G_{i-1}) - cc(G_i) < \epsilon p \cdot \gamma n/16$ (and $\mathcal{Y}_i = 0$ otherwise), which is the event that the edges of $M_p(i)$ do not reduce the number of connected components in G_{i-1} by more than $\epsilon p \cdot \gamma n/16$. Suppose for at least half of the rounds, $\mathcal{Y}_i = 0$; then, the number of connected components after the IMSP selects the r matchings is less than

$$n - \frac{r}{2} \cdot \frac{\epsilon p \cdot \gamma n}{16} \leq n - \frac{32 \log(2e/\gamma)}{\epsilon p} \cdot \frac{1}{2} \cdot \frac{\epsilon p \cdot \gamma n}{16} \leq (1 - \gamma)n.$$

On the other hand, since having $|U| + (1 - \gamma + \epsilon\gamma)n$ odd-size connected components (i.e., $\mathcal{E}_U$ happens) implies that there are more than $(1 - \gamma)n$ connected components,

$$\Pr[\mathcal{E}_U] = \Pr\Big(\mathcal{E}_U, \sum_{i \in [r]} \mathcal{Y}_i \geq \frac{r}{2}\Big) \tag{3.2}$$

Hence, we can focus on upper bounding the probability that $\mathcal{E}_U$ happens *and* for more than half of the rounds, $\mathcal{Y}_i = 1$. In the following, we first establish a key property regarding the probability that $\mathcal{Y}_i = 1$ (Lemma 3.4.4) and then show how to use this property to bound the probability of the target event in Eq (3.2) (Lemma 3.4.6). To simplify the presentation, we use $M_p(i_i, i_2, \ldots, i_j)$ to denote the set of edges realized in the matchings $M_p(i_1), M_p(i_2), \ldots, M_p(i_j)$, and, with a slight abuse of notation, use $\mathcal{Y}_i$ to denote the event that $\mathcal{Y}_i = 1$. In particular, we show that

Lemma 3.4.4. *For any $M_p(1, \ldots, i-1)$,*

$$\Pr\left(\mathcal{Y}_i, \mathcal{E}_{i-1} \mid M_p(1, \ldots, i-1)\right) \leq \exp\left(-\frac{3\epsilon p \cdot \gamma n}{16}\right)$$

Proof. Recall that $\mathcal{E}_{i-1}$ is the event that the number of odd-size connected components in G_{i-1} is at least $|U| + (1 - \gamma + \epsilon\gamma)n$. We have,

$$\begin{aligned}
&\Pr[\mathcal{Y}_i, \mathcal{E}_{i-1} \mid M_p(1, \ldots, i-1)] \\
=&\Pr[\mathcal{Y}_i \mid M_p(1, \ldots, i-1), \mathcal{E}_{i-1}] \cdot \Pr[\mathcal{E}_{i-1} \mid M_p(1, \ldots, i-1)] && \text{(Chain rule)} \\
\leq&\Pr[\mathcal{Y}_i \mid M_p(1, \ldots, i-1), \mathcal{E}_{i-1}]
\end{aligned}$$

hence, we only need to show that $\Pr[\mathcal{Y}_i \mid M_p(1, \ldots, i-1), \mathcal{E}_{i-1}] \leq \exp\left(-\frac{3\epsilon p \cdot \gamma n}{16}\right)$, which is, roughly speaking, the probability that the i-th matching M_i does not reduce the number of connected components by a lot, given that the graph G_{i-1} contains many odd-size connected components.

To proceed, we need the following definition. For any graph H, we say a set of edges E' form a *component-based spanning forest* of H, if E' is a spanning forest of the graph obtained by contracting each connected component in H into a single vertex (and any edge $(u, v) \in E'$ becomes an edge between the connected components that u and v respectively resides in). It is straightforward to verify that if we add any component-based spanning forest $E' \subset E$ to H, the number of connected components in H would reduce by $|E'|$. The following claim is the key to obtain the target upper bound on $\Pr[\mathcal{Y}_i \mid M_p(1, \ldots, i-1), \mathcal{E}_{i-1}]$.

Claim 3.4.5. *Whenever $\mathcal{E}_{i-1}$ happens, there exist at least $\frac{\epsilon\gamma n}{2}$ edges of M_i that form a component-based spanning forest of G_{i-1}.*

Proof. Since the matching M_i has size at least $\gamma n/2$ (by definition of being in a γ-matching-cover), the edges of M_i can reduce the number of odd-size sets from at least $|U|+(1-\gamma+\epsilon\gamma)n$ (i.e., $\mathcal{E}_{i-1}$ happens) down to at most $|U| + (1 - \gamma)n$ (by Lemma 3.3.1). Therefore, after adding edges of M_i to G_{i-1}, at least $\epsilon\gamma n$ odd-size sets will disappear. For each odd-size set S that disappears, M_i must contain at least one edge between S and another connected component. Therefore, for the largest component-based spanning forest obtained by the

edges in M_i, at least $\epsilon\gamma n$ vertices (i.e., connected components) have degree at least one, which implies the number of edges in the forest is at least $\frac{\epsilon\gamma n}{2}$. □

Let $T_i \subseteq M_i$ be the the set of (at least) $\frac{\epsilon\gamma n}{2}$ edges promised by Claim 3.4.5 (conditioned on $\mathcal{E}_{i-1}$) and t_i be the number of edges realized in T_i. By Claim 3.4.1, each edge in T_i is realized w.p. at least p independent of each other, hence, $\mathrm{E}[t_i \mid \mathcal{E}_{i-1}] \geq \epsilon p \cdot \gamma n/2$. Note that t_i is a lower bound on $cc(G_{i-1}) - cc(G_i)$ (i.e., the decrement of the number of connected components from G_{i-1} to G_i), and $\mathcal{Y}_i = 1$ iff $cc(G_{i-1}) - cc(G_i) \leq \epsilon p \cdot \gamma n/16$, which implies that no more than $\epsilon p \cdot \gamma n/16$ edges is realized in T_i (which is 1/8 of the expectation). Hence, by Chernoff bound,

$$\begin{aligned}
&\Pr[\mathcal{Y}_i \mid M_p(1,\ldots,i-1), \mathcal{E}_{i-1}] \leq \Pr[t_i \leq \frac{\epsilon p \cdot \gamma n}{16} \mid \mathcal{E}_{i-1}] \\
&\leq \exp\left(-\frac{1}{2} \cdot \left(\frac{7}{8}\right)^2 \cdot \frac{\epsilon p \cdot \gamma n}{2}\right) = \exp\left(-\frac{49}{64} \cdot \frac{\epsilon p \cdot \gamma n}{4}\right) \\
&\leq \exp\left(-\frac{48}{64} \cdot \frac{\epsilon p \cdot \gamma n}{4}\right) \leq \exp\left(-\frac{3\epsilon p \cdot \gamma n}{16}\right)
\end{aligned}$$

which concludes the proof of Lemma 3.4.4. □ □

Having Lemma 3.4.4, we are now ready to upper bound the probability that $\mathcal{Y}_i$ happens in more than half of the rounds.

Lemma 3.4.6. *For any collection of $r/2$ rounds $i_1, i_2, \ldots, i_{r/2}$,*

$$Pr[\mathcal{Y}_{i_1}, \mathcal{Y}_{i_2}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_U] \leq 2^{-3\gamma n \log(2e/\gamma)}.$$

Proof. Assume wlog that $i_1 < i_2 < \ldots < i_{r/2}$. First of all, $\mathcal{E}_U$ happening implies that $\mathcal{E}_{i_1-1}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{E}_{i_{r/2}-1}$ should all happen[4]; therefore,

$$\Pr[\mathcal{Y}_{i_1}, \mathcal{Y}_{i_2}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_U] \leq \Pr[\mathcal{Y}_{i_1}, \mathcal{Y}_{i_2}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_{i_1-1}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{E}_{i_{r/2}-1}] \tag{3.3}$$

[4]It is straightforward to very that the number of odd-size connected components is monotonically decreasing.

After reorganizing the terms, we have,

$$\begin{aligned}
&\Pr[\mathcal{Y}_{i_1}, \mathcal{Y}_{i_2}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_{i_1-1}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{E}_{i_{r/2}-1}] \\
=&\Pr[\mathcal{Y}_{i_1}, \mathcal{E}_{i_1-1}, \mathcal{Y}_{i_2}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_{i_{r/2}-1}] \\
=&\Pi_{j\in[r/2]}\Pr[\mathcal{Y}_{i_j}, \mathcal{E}_{i_j-1} \mid \mathcal{Y}_{i_1}, \mathcal{E}_{i_1-1}, \mathcal{Y}_{i_2}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{Y}_{i_{j-1}}, \mathcal{E}_{i_{j-1}-1}] \qquad \text{(Chain rule)}
\end{aligned}$$

We will upper bound each of the $r/2$ terms separately. Fix a $j \in [r/2]$, denote the event $(\mathcal{Y}_{i_1}, \mathcal{E}_{i_1-1}, \mathcal{Y}_{i_2}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{Y}_{i_{j-1}}, \mathcal{E}_{i_{j-1}-1})$ by $\mathcal{E}^*$. Note that $\mathcal{E}^*$ is completely determined by $M_p(1, \ldots, i_{j-1})$, which is also determined by $M_p(1, \ldots, i_j - 1)$ since $i_j - 1 \geq i_{j-1}$. We have

$$\begin{aligned}
&\Pr[\mathcal{Y}_{i_j}, \mathcal{E}_{i_j-1} \mid \mathcal{Y}_{i_1}, \mathcal{E}_{i_1-1}, \mathcal{Y}_{i_2}, \mathcal{E}_{i_2-1}, \ldots, \mathcal{Y}_{i_{j-1}}, \mathcal{E}_{i_{j-1}-1}] = \Pr[\mathcal{Y}_{i_j}, \mathcal{E}_{i_j-1} \mid \mathcal{E}^*] \\
=&\sum_{M_p(1,\ldots,i_j-1)} \Pr[M_p(1, \ldots, i_j - 1) \mid \mathcal{E}^*] \cdot \Pr[\mathcal{Y}_{i_j}, \mathcal{E}_{i_j-1} \mid \mathcal{E}^*, M_p(1, \ldots, i_j - 1)] \\
=&\sum_{M_p(1,\ldots,i_j-1) \text{ s.t. } \mathcal{E}^* happens} \Pr[M_p(1, \ldots, i_j - 1) \mid \mathcal{E}^*] \cdot \Pr[\mathcal{Y}_{i_j}, \mathcal{E}_{i_j-1} \mid M_p(1, \ldots, i_j - 1)] \\
&\qquad\qquad (\mathcal{E}^* \text{ is determined by } M_p(1, \ldots, i_j - 1)) \\
\leq&\sum_{M_p(1,\ldots,i_j-1) \text{ s.t. } \mathcal{E}^* happens} \Pr[M_p(1, \ldots, i_j - 1) \mid \mathcal{E}^*] \cdot \exp\left(-\frac{3\epsilon p \cdot \gamma n}{16}\right) \quad \text{(By Lemma 3.4.4)} \\
=&\exp\left(-\frac{3\epsilon p \cdot \gamma n}{16}\right)
\end{aligned}$$

Therefore

$$\Pr[\mathcal{Y}_{i_1}, \mathcal{Y}_{i_2}, \ldots, \mathcal{Y}_{i_{r/2}}, \mathcal{E}_U] \leq \exp\Big(-\frac{3\epsilon p \cdot \gamma n}{16} \cdot \frac{r}{2}\Big) \leq \exp(-3\gamma n \log(2e/\gamma)) \leq 2^{-3\gamma n \log(2e/\gamma)}$$

where the second equality is by the choice of r. □

By Lemma 3.4.6, for each collection of $\frac{r}{2}$ rounds, $\mathcal{Y}_i$ happens to all of them w.p. at most $2^{-3\gamma n \log(2e/\gamma)}$. There are at most 2^r (which is independent of n) choices of different (at least) $\frac{r}{2}$ rounds, hence using union bounds, for n sufficiently large, the prob. that $\mathcal{Y}_i$ happens in more than $\frac{r}{2}$ rounds is at most $2^{-2\gamma n \log(2e/\gamma)}$, proving Lemma 3.4.3. As discussed earlier,

Lemma 3.4.3 implies Lemma 3.4.2, which completes the proof. □ □

□

3.4.2. Vertex Sparsification Lemma

In the following, we give an algorithm that for any $0 < \epsilon < 1$, reduces the number of vertices in any graph G from $|V|$ to $O(\text{opt}(G)/\epsilon)$, while preserving the maximum matching size to within a factor of $(1-\epsilon)$ for *any* realization G_p of G w.h.p.

For inputs G and ϵ and a sparsification parameter τ to be determined later, $\mathsf{SPARSIFY}(G, \tau, \epsilon)$ works as follows. We create and output a multi-graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ where: (i) $|\mathcal{V}| = \tau$, (ii) each vertex v in G is mapped to a vertex $\mathcal{V}(v)$ in $\mathcal{G}$ chosen *uniformly at random* from $\mathcal{V}$, and (iii) for each edge (u, v), there is a corresponding edge between $\mathcal{V}(u)$ and $\mathcal{V}(v)$. A pseudo-code of the SPARSIFY algorithm is presented in Algorithm 1. We point out that similar ideas of randomly grouping vertices for matchings have been also recently used in [16, 31] for the purpose of reducing *space requirement* of algorithms in graph streams.

Algorithm 1: $\mathsf{SPARSIFY}(G, \tau, \epsilon)$. A Matching-Preserving Sparsification Algorithm

Input: Graph $G(V, E)$, sparsification parameter τ, and input parameter $\epsilon > 0$.
Output: A multi-graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ with $|\mathcal{V}| = \tau$.

1. Partition the vertices in V into τ groups $\mathcal{V} := (\mathcal{V}_1, \ldots, \mathcal{V}_\tau)$, by assigning each vertex *independently* to one of the τ groups chosen *uniformly at random*.
2. For any edge $(u, v) \in E$, add an edge $e_{u,v}$ between the vertices $\mathcal{V}(u)$ and $\mathcal{V}(v)$ in $\mathcal{E}$.
3. Return the multi-graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$.

The following lemma states the main property of the SPARSIFY algorithm.

Lemma 3.4.7. *For any graph $G(V, E)$, suppose $\mathcal{G}(\mathcal{V}, \mathcal{E}) := \mathsf{SPARSIFY}(G, \tau, \epsilon)$ for the parameter $\tau \geq \frac{4 \cdot opt(G)}{\epsilon}$, and let M be any fixed matching in G with $|M| = \omega(1)$; then, w.p. $1 - o(1)$, there exists a matching $\mathcal{M}$ of size $(1-\epsilon) \cdot |M|$ in $\mathcal{G}$. Moreover, the edges of $\mathcal{M}$ correspond to a unique matching M' in G of the same size.*

Proof. Let $\epsilon' := \epsilon/4$, and let s denote the number of vertices matched in M (i.e., $s = 2|M|$).

Note that $\tau \geq 4\mathrm{opt}(G)/\epsilon \geq 4s/\epsilon = s/\epsilon'$. For the sake of analysis, we first merge the τ groups $\mathcal{V}$ into s/ϵ' *super-groups* in the following manner. Fix any partition that evenly breaks $[\tau]$ into s/ϵ' non-empty parts $P_1, P_2, \ldots, P_{s/\epsilon'}$ (i.e., $|P_i| = \frac{\epsilon'\tau}{s}$ for any $i \in [s/\epsilon']$). For each partition P_i, define a super-group $\mathcal{S}_i := \cup_{j\in[P_i]}\mathcal{V}_j$. Then, since in Algorithm 1 each vertex $v \in V$ is assigned to exactly one group chosen uniformly at random, the probability that $v \in \mathcal{S}_i$ is $\frac{\epsilon'}{s}$.

We say a super-group $\mathcal{S}_i$ is *good* iff $\mathcal{S}_i$ contains at least one vertex in $V(M)$, and is otherwise *bad*. For each super-group $\mathcal{S}_i$, let X_i be a random variable where $X_i = 1$ iff $\mathcal{S}_i$ is bad (otherwise, $X_i = 0$). Let $X := \sum_{i\in[s/\epsilon']} X_i$ be the number of bad super-groups. In the following, we first show that X is small w.h.p., which implies that there are many good super-groups, and then show that there is a large matching between the good super-groups.

To see that $X = \sum_{i\in[s/\epsilon']} X_i$ is small w.h.p., first notice that

$$\Pr(X_i = 1) = \left(1 - \frac{\epsilon'}{s}\right)^s \leq e^{-\epsilon'} \leq 1 - \epsilon' + \frac{\epsilon'^2}{2} \qquad (\forall x \geq 0, e^{-x} \leq 1 - x + x^2/2)$$

Therefore, we have $\mathrm{E}[X] = \sum_{i\in[s/\epsilon']} \mathrm{E}[X_i] \leq (1 - \epsilon' + \frac{\epsilon'^2}{2})\frac{s}{\epsilon'}$. On the other hand, since at most $|V(M)|\,(= s)$ super-groups could contain a vertex from $V(M)$ (i.e., could be good), $X \geq s/\epsilon' - s = \Omega(s) = \omega(1)$, and hence $\mathrm{E}[X] = \omega(1)$. To continue, observe that our setting can be viewed as a standard balls and bins experiment: each vertex in $V(M)$ is a ball; each super-group is a bin; and X_i denotes the event that the i-th bin is empty. Therefore, the random variables X_i's are *negatively correlated*, and we can apply Chernoff bound [92]:

$$\Pr\left(X \geq (1+\epsilon'^2)\mathrm{E}[X]\right) \leq e^{-\Omega_{\epsilon'}(\mathrm{E}[X])} = o(1) \qquad (\mathrm{E}[X] = \omega(1))$$

Since $\mathrm{E}[X] \leq (1 - \epsilon' + \frac{\epsilon'^2}{2})\frac{s}{\epsilon'}$ (as shown above), we further have

$$\Pr[X \geq (1+\epsilon'^2)(1 - \epsilon' + \frac{\epsilon'^2}{2})\frac{s}{\epsilon'}] \leq \Pr\left(X \geq (1+\epsilon'^2)\mathrm{E}[X]\right) = o(1)$$

Hence, w.p. $1-o(1)$, the number of bad super-groups is at most $(1-\epsilon'+2\epsilon'^2)\frac{s}{\epsilon'}$, which implies that the number of good super-groups is at least $\frac{s}{\epsilon'}-(1-\epsilon'+2\epsilon'^2)\frac{s}{\epsilon'} = (\epsilon'-2\epsilon'^2)\frac{s}{\epsilon'} = (1-2\epsilon')s$.

It remains to show that if at least $(1-2\epsilon')s$ super-groups are good (i.e., contain a vertex from $V(M)$), then the edges in M form a matching $\mathcal{M}$ in $\mathcal{G}$ of size at least $(1-4\epsilon')\,|M|$ $(= (1-\epsilon)\,|M|)$. To see this, for each good super-group $\mathcal{S}_i$, we fix one vertex $v \in V(M) \cap \mathcal{S}_i$ and remove all other vertices in $\mathcal{S}_i$. For the matching M, at most $2\epsilon' s$ vertices in $V(M)$ are removed and hence at least $s/2 - 2\epsilon' s = (1-4\epsilon')\,|M|$ edges in M remain. Since all endpoints of these edges are assigned to distinct super-groups, these edges form a matching $\mathcal{M}$ of size at least $(1-4\epsilon')\,|M| = (1-\epsilon)\,|M|$ in $\mathcal{G}$.

To see the second part of the lemma, simply note that each edge of $\mathcal{M}$ comes from a distinct edge in M. □

For any $G_p := G(V, E_p)$, define $\mathcal{G}_p$ as the graph obtained from $\mathcal{G}$ by considering only the edges that correspond to edges in E_p. We are now ready to prove our vertex sparsification lemma.

Lemma 3.4.8 (Vertex Sparsification Lemma)**.** *Let $G(V,E)$ be a graph with maximum matching size $\omega(1/p)$ and let $\mathcal{G} = \mathsf{SPARSIFY}(G, \tau, \epsilon)$ for $\tau = \frac{4\cdot opt(G)}{\epsilon}$; then,*

$$\Pr\Big(opt(\mathcal{G}_p) \geq (1-\epsilon)\cdot opt(G_p)\Big) = 1 - o(1)$$

where the probability is taken over both the inner randomness of SPARSIFY *algorithm as well as the realization $E_p \subseteq E$.*

Proof. By Claim 3.3.2, the maximum matching size in G_p is $\omega(1)$ w.p. $1-o(1)$. Now, for any realization G_p with maximum matching size of $\omega(1)$, by construction, $\mathcal{G}_p = \mathsf{SPARSIFY}(G_p, \tau, \epsilon)$. Hence, by applying Lemma 3.4.7 on any maximum matching M in G_p, we have that $\mathcal{G}_p$ has a matching of size $(1-\epsilon)\,|M|$ w.p. $1-o(1)$. □

3.5. A $(1-\epsilon)$-Approximation Adaptive Algorithm

We now present our adaptive algorithm and prove Theorem 3.1. The following is a formal restatement of Theorem 3.1.

Theorem 3.3. *There is an* adaptive *algorithm that for any input graph $G(V,E)$, and any input parameter $\epsilon > 0$, outputs a matching of size $ALG := ALG(G_p)$ such that,*

$$\Pr\Big(ALG \geq (1-\epsilon)\cdot opt\Big) = 1 - o(1)$$

where $opt := opt(G_p)$ is the maximum matching size in $G_p(V,E_p)$. The probability is taken over both the inner randomness of the algorithm and the realization $E_p \subseteq E$.

Moreover, the algorithm makes only $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ rounds of adaptive queries, and queries only $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ edges per vertex.

Our adaptive algorithm in Theorem 3.3 works as follows. We first use our vertex sparsification lemma (Lemma 3.4.8) to compute a graph $\mathcal{G}(\mathcal{V},\mathcal{E}) := \mathsf{SPARSIFY}(G,\tau,\epsilon)$ where $\text{opt}(\mathcal{G}) = \Omega(|\mathcal{V}|)$. Next, we repeat for $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ rounds the following operation. Pick a maximum matching M from $\mathcal{G}$, query the edges of M, and remove the edges that are not realized. Finally, return a maximum matching among the realized edges. The pseudo-code of this algorithm is presented as Algorithm 2.

Algorithm 2: A $(1-\epsilon)$-Approximation Adaptive Algorithm for Stochastic Matching

Input: Graph $G(V,E)$ and input parameters $0 < \epsilon, p < 1$.
Output: A matching M in $G(V,E_p)$.

1. Let $\mathcal{G}(\mathcal{V},\mathcal{E}) := \mathsf{SPARSIFY}(G,\tau,\epsilon)$ for $\tau := \left\lceil \frac{4\text{opt}(G)}{\epsilon} \right\rceil$.
2. Let $R := \left\lceil \frac{32\log(8e/\epsilon p)}{\epsilon\cdot p} \right\rceil$, and $\mathcal{E}^* \leftarrow \mathcal{E}$.
3. For $i = 1,\ldots,R$, do:
   (a) Pick a *maximum matching* M_i in $\mathcal{G}(\mathcal{V},\mathcal{E}^*)$.
   (b) *Query* the edges in M_i and remove the *non-realized edges* from $\mathcal{E}^*$.
4. Output a maximum matching among *realized* edges in $M_1, M_2, \ldots, M_R$.

It is straightforward to verify the number of queries and the degree of adaptivity used in

Algorithm 2: each round of the algorithm queries edges of a matching and hence each vertex is queried at most once in each round. We now prove the bound on the approximation ratio of the algorithm.

Proof of Theorem 3.3. Let $\widehat{\text{opt}} := \widehat{\text{opt}}(\mathcal{G}_p)$ denote the maximum matching size in the graph $\mathcal{G}_p$. By Lemma 3.4.8, w.p. $1-o(1)$, $\widehat{\text{opt}} \geq (1-\epsilon)\text{opt}(G_p)$. Now, it suffices to show that w.p. $1-o(1)$, Algorithm 2 outputs a matching of size at least $(1-\epsilon)\widehat{\text{opt}}$, since, with a union bound, it would imply w.p. $1-o(1)$,

$$\text{ALG}(G_p) \geq (1-\epsilon)\cdot\widehat{\text{opt}} \geq (1-\epsilon)^2\cdot\text{opt}(G_p) \geq (1-2\epsilon)\cdot\text{opt}(G_p)$$

and we can replace ϵ with $\epsilon/2$ in Algorithm 2 to obtain a $(1-\epsilon)$-approximation.

To see that $\text{ALG}(G_p) \geq (1-\epsilon)\widehat{\text{opt}}$ w.h.p., let $L := \min_{i\in[R]}|M_i|$, i.e., the minimum size of a matching chosen by Algorithm 2. Since all M_i's are maximum matchings in $\mathcal{E}^*$ while $\mathcal{E}^*$ always contains all edges of the optimum matching in $\mathcal{G}_p$, we have $L \geq \widehat{\text{opt}}$ and we can focus on showing $\text{ALG}(G_p) \geq (1-\epsilon)L$.

It is straightforward to verify that the way Algorithm 2 selects the matchings $M_1, M_2, \ldots, M_R$ satisfies the condition of IMSP (Definition 3.1). Moreover, by Claim 3.3.2 and Lemma 3.4.8, we have, $\Pr\Big(\widehat{\text{opt}} \geq p\cdot\text{opt}(G)/2\Big) = 1-o(1)$. Therefore, we can use Lemma 3.4.2 with parameters:

$$\gamma = \frac{2L}{|\mathcal{V}|} \geq \frac{\epsilon\cdot p}{4} \qquad , \qquad r = \frac{32\log(2e/\gamma)}{\epsilon p} \leq \frac{32\log(8e/\epsilon p)}{\epsilon p} \leq R$$

which states that w.p. $1-o(1)$, the realized edges in matchings $M_1, M_2, \ldots, M_R$ contain a matching of size at least $(1-\epsilon)\frac{\gamma|\mathcal{V}|}{2} = (1-\epsilon)L$, which completes the proof. □

3.6. A $\left(\frac{1}{2}-\epsilon\right)$-Approximation Non-Adaptive Algorithm

In this section, we present our non-adaptive algorithm and prove Theorem 3.2. The following is a formal restatement of Theorem 3.2.

Theorem 3.4. *There is a* non-adaptive *algorithm that for any input graph $G(V,E)$, any input parameter $\epsilon > 0$, outputs a matching of size $ALG := ALG(G_p)$ such that,*

$$\Pr\Big(ALG \geq (\frac{1}{2}-\epsilon)\cdot opt\Big) = 1 - o(1)$$

where $opt := opt(G_p)$ is the maximum matching size in $G_p(V,E_p)$. The probability is taken over both the inner randomness of the algorithm and the realization $E_p \subseteq E$.

Moreover, the algorithm non-adaptively queries $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ edges per vertex.

Note that any non-adaptive algorithm works in the following framework:

1. Compute a subgraph $H(V,Q)$ of the input graph $G(V,E)$ for some $Q \subseteq E$.
2. Query all edges in Q and compute a maximum matching in $H(V,Q_p)$.

Therefore, the main task of any non-adaptive algorithm is to choose a "good" subgraph H. Our non-adaptive algorithm chooses a subgraph H as follows. We first use our vertex sparsification lemma to compute a graph $\mathcal{G}(\mathcal{V},\mathcal{E}) := \mathsf{SPARSIFY}(G,\tau,\epsilon)$ where $\text{opt}(\mathcal{G}) = \Omega(|\mathcal{V}|)$. Next, we repeat for $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ times the process of picking a maximum matching from $\mathcal{G}(\mathcal{V},\mathcal{E})$ and removing the edges of the matching from $\mathcal{E}$. Let Q be the set of edges in these matchings; the algorithm returns $H(V,Q)$ as the subgraph H. A pseudo-code of this algorithm is presented as Algorithm 3.

We now briefly provide the intuition behind Algorithm 3. Similar to the adaptive case, using the vertex sparsification lemma (Lemma 3.4.8), our task reduces to approximating the maximum matching in $\mathcal{G}_p$ (as opposed to G_p). For the adaptive case, our algorithm guarantees that every selected matching is of size at least $\text{opt}(\mathcal{G}_p)$, which allows us to use

Algorithm 3: A Non-Adaptive $\left(\frac{1}{2}-\epsilon\right)$-Approximation Algorithm for Stochastic Matching

Input: Graph $G(V,E)$ and input parameters $0 < \epsilon, p < 1$.
Output: A matching M in $G(V, E_p)$.

1. Let $\mathcal{G}(\mathcal{V},\mathcal{E}) := \mathsf{SPARSIFY}(G, \tau, \epsilon)$ for $\tau := \left\lceil \frac{4\text{opt}(G)}{\epsilon} \right\rceil$.
2. Let $R := \left\lceil \frac{32\log(16e/\epsilon p)}{\epsilon \cdot p} \right\rceil$.
3. Initially $Q \leftarrow \emptyset$. For $i = 1, \ldots, R$, do:
   (a) Pick a *maximum matching* M_i in $\mathcal{G}(\mathcal{V}, \mathcal{E} \setminus Q)$.
   (b) Let $Q \leftarrow Q \cup M_i$.
4. *Query* all edges in Q and return a maximum matching in $M_1, M_2, \ldots, M_R$.

the matching-cover lemma directly to complete the argument. However, Algorithm 3 does not have such a strong guarantee since it is non-adaptive. To address this issue, we establish a weaker guarantee which allows us to obtain a 1/2-approximation. The idea is as follows. On one hand, if the smallest matching selected by the algorithm is of size at least $\text{opt}(\mathcal{G}_p)/2$, we can still invoke the matching-cover lemma (Lemma 3.4.2) and have that Algorithm 3 outputs a matching of size $(1-\epsilon)\text{opt}(\mathcal{G}_p)/2$. On the other hand, we show that if the smallest selected matching has size less than $\text{opt}(\mathcal{G}_p)/2$, then for *any* maximum matching M^* in $\mathcal{G}_p$, we must have selected in Q at least half the edges of M^*, which immediately results in a matching of size $\text{opt}(\mathcal{G}_p)/2$.

We now present the formal proof. In the following, let $L := \min_{i\in[R]} |M_i|$, i.e., the minimum size of a matching chosen by Algorithm 3. Note that $L = |M_R|$ since the size of matchings chosen by the algorithm is a *non-increasing* sequence by construction.

Lemma 3.6.1. *If $L \geq p \cdot opt(G)/4$, then,* $\Pr\Big(ALG \geq (1-\epsilon)\cdot L\Big) = 1 - o(1)$.

Proof. Since $M_1, \ldots, M_R$ are edge disjoint matchings, the process of choosing $M_1, \ldots, M_R$ is an IMSP (Definition 3.1). Hence, by Lemma 3.4.2 with parameters:

$$\gamma = \frac{2L}{|\mathcal{V}|} \geq \frac{\epsilon \cdot p}{8} \qquad , \qquad r = \frac{32\log(2e/\gamma)}{\epsilon p} \leq \frac{32\log(16e/\epsilon p)}{\epsilon p} = R$$

there exists a matching of size $(1-\epsilon)\cdot\frac{\gamma|\mathcal{V}|}{2} = (1-\epsilon)\cdot L$ in Q_p w.p. $1 - o(1)$. Noting that any matching of $\mathcal{G}_p$ in Q_p corresponds to a matching in G_p completes the proof. □

We define $\widehat{\text{opt}} := \widehat{\text{opt}}(\mathcal{G}_p)$ as the maximum matching size in $\mathcal{G}_p$.

Lemma 3.6.2. $\Pr\Big(ALG \geq \left(\frac{1}{2} - \epsilon\right) \cdot \widehat{opt} \mid \widehat{opt} > 2L\Big) = 1.$

Proof. Let M be any arbitrary matching in $\mathcal{G}$. We have, $|M| - |Q \cap M| \leq L$ since otherwise, for the matching $M' := M \setminus Q$, we have $|M'| = |M| - |Q \cap M| > L > |M_R|$, contradicting the fact that M_R is a maximum matching in the remaining graph.

Now let M be any maximum matching in $\mathcal{G}_p$. Since $\widehat{\text{opt}} > 2L$, we have $|M| > 2L$. Consequently, $|Q \cap M| \geq |M| - L \geq |M| - |M|/2 = |M|/2$. Hence, at least half of the edges in M are also present in Q, implying that in this case, $\text{ALG} \geq \widehat{\text{opt}}/2$ w.p. 1. □

We now prove the bound on the approximation ratio of Algorithm 3.

Lemma 3.6.3. $\Pr\Big(ALG \geq (\frac{1}{2} - 2\epsilon) \cdot opt\Big) = 1 - o(1).$

Proof. Recall that $\widehat{\text{opt}} := \widehat{\text{opt}}(\mathcal{G}_p)$. By Claim 3.3.2 and Lemma 3.4.8, we have,

$$\Pr\Big(\widehat{\text{opt}} \geq p \cdot \text{opt}(G)/2\Big) = 1 - o(1) \tag{3.4}$$

Let $\mathcal{E}win$ be the event that $\text{ALG} \geq \left(\frac{1}{2} - \epsilon\right) \cdot \widehat{\text{opt}}$. We argue that $\Pr(\mathcal{E}win) = 1 - o(1)$. This, together with the fact that w.p. $1 - o(1)$, $\widehat{\text{opt}} \geq (1 - \epsilon) \cdot \text{opt}$ (Lemma 3.4.8) completes the proof.

Consider two cases, (i) $L < p \cdot \text{opt}(G)/4$, and (ii) $L \geq p \cdot \text{opt}(G)/4$. For case (i),

$$\begin{aligned}\Pr(\mathcal{E}win) &\geq \Pr\left(\widehat{\text{opt}} > 2L\right) \cdot \Pr\left(\mathcal{E}win \mid \widehat{\text{opt}} > 2L\right) \\ &\geq \Pr\left(\widehat{\text{opt}} > p \cdot \text{opt}(G)/2\right) \cdot \Pr\left(\mathcal{E}win \mid \widehat{\text{opt}} > 2L\right) = (1 - o(1)) \cdot 1\end{aligned}$$

where the last equality is by Eq (3.4) and Lemma 3.6.2. For case (ii),

$$\begin{aligned}
\Pr(\mathcal{E}win) =& \Pr\left(\widehat{\text{opt}} > 2L\right) \cdot \Pr\left(\mathcal{E}win \mid \widehat{\text{opt}} > 2L\right) + \Pr\left(\widehat{\text{opt}} \leq 2L\right) \cdot \Pr\left(\mathcal{E}win \mid \widehat{\text{opt}} \leq 2L\right) \\
=& \Pr\left(\widehat{\text{opt}} > 2L\right) + \Pr\left(\widehat{\text{opt}} \leq 2L\right) \cdot \Pr\left(\mathcal{E}win \mid \widehat{\text{opt}} \leq 2L\right) \qquad \text{(By Lemma 3.6.2)} \\
\geq& \Pr\left(\widehat{\text{opt}} > 2L\right) + \Pr\left(\widehat{\text{opt}} \leq 2L\right) \cdot \Pr\left(\text{ALG} \geq (1-\epsilon) \cdot L \mid \widehat{\text{opt}} \leq 2L\right) \\
\geq& \Pr\left(\widehat{\text{opt}} > 2L\right) \cdot \Pr\left(\text{ALG} \geq (1-\epsilon) \cdot L \mid \widehat{\text{opt}} > 2L\right) \\
& \qquad + \Pr\left(\widehat{\text{opt}} \leq 2L\right) \cdot \Pr\left(\text{ALG} \geq (1-\epsilon) \cdot L \mid \widehat{\text{opt}} \leq 2L\right) \\
=& \Pr(\text{ALG} \geq (1-\epsilon) \cdot L) = 1 - o(1)
\end{aligned}$$

where the last equality is by Lemma 3.6.1. □

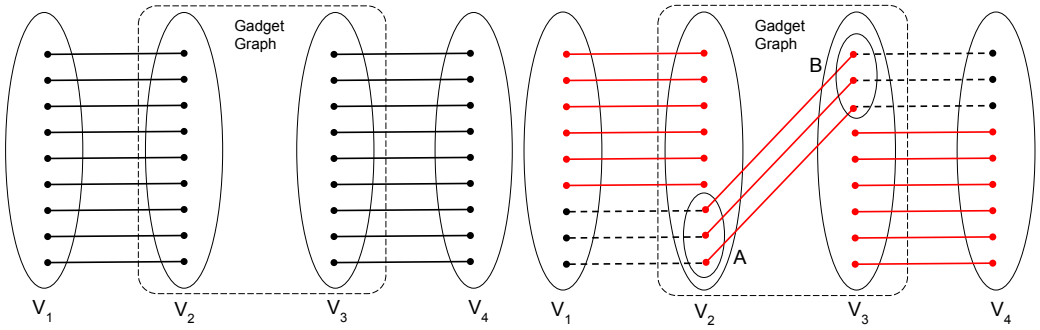
Theorem 3.4 now follows from Lemma 3.6.3 (by replacing ϵ with $\epsilon/2$ in Algorithm 3), and the fact that Algorithm 3 queries $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ matchings and hence $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ incident edges are queried for each vertex.

3.7. A Barrier to Obtaining a Non-Adaptive $(1-\epsilon)$-Approximation Algorithm

The approximation ratio of our non-adaptive algorithm is $(\frac{1}{2} - \epsilon)$ as opposed to the near-optimal ratio of $(1-\epsilon)$ achieved by our adaptive algorithm. A natural question, first raised by [25], is if one can obtain a $(1-\epsilon)$-approximation using a non-adaptive algorithm, even by allowing arbitrary dependence on p and ϵ. In the following, we highlight a possible barrier to obtain such a result.

Consider a bipartite graph $G(L, R, E)$ constructed as follows: (i) the vertex sets are $L = V_1 \cup V_3$, $R = V_2 \cup V_4$ and $|V_i| = N$ for $i \in [4]$, (ii) there is a perfect matching between V_1 and V_2, and a perfect matching between V_3 and V_4, and (iii) there is a *gadget graph* $\widehat{G}(V_2, V_3, \widehat{E})$, to be determined later, between V_2 and V_3 (see Fig 3.a).

Suppose we want to design a non-adaptive $(1-\epsilon)$-approximation algorithm for the instance $G(V, E)$ with the parameter $p = 2/3$. In this case, for any graph G_p, w.h.p., there is a



(a) Input graph $G(V, E)$ (b) A realization $G_p(V, E_p)$

Figure 3: An example of a barrier to $(1-\epsilon)$-approximation non-adaptive algorithms. The edges in the gadget graph are *not* presented in this figure. In part (b), solid red edges (resp. dashed edges) are the edges that are realized (resp. not realized).

matching M_1 between V_1 and V_2, and another matching M_2 between V_3 and V_4, each of size $(2/3)N - o(N)$. Hence,

$$\text{opt}(G_p) \geq |M_1| + |M_2| + (2/3) \cdot m(A, B) - o(N) \tag{3.5}$$

where A (resp. B) is the set of vertices in V_2 (resp. V_3) that are not matched by M_1 (resp. M_2), and $m(A, B)$ denotes the size of a maximum matching between A and B. A few observations are in order. First, picking edges of M_1 and M_2 is crucial for having any large matching in G_p, and second, for a uniformly at random chosen realization of M_1 and M_2, the set A and B are chosen uniformly at random from V_2 and V_3 (see Fig 3.b). Based on these observations, we define the following problem.

Problem 1. *Given a bipartite graph $G(L, R, E)$, choose a subgraph $H(L, R, Q)$ such that given two subsets $A \subseteq L$ and $B \subseteq R$, if $m(A, B) \geq N/3 - o(N)$ in G, then H contains* at least $\Omega(N)$ *edges between A and B.*

The goal is to solve Problem 1 using a graph H with small number of edges. The previous discussion implies that any non-adaptive $(1-\epsilon)$-approximation algorithm has to solve Problem 1 for the gadget graph $\widehat{G}$, when the two sets A and B are chosen *uniformly at random*. Otherwise, for the maximum matching size $\text{ALG}(G_p)$ in $H(V, Q_p)$ and maximum

matching size $\text{OPT}(G_p)$ in G_p, we have:

$$\text{ALG}(G_p) \leq |M_1| + |M_2| + o(N) \leq (4/3)N + o(N)$$
$$\text{OPT}(G_p) \geq (4/3)N + (2/3)(N/3) - o(N) = (14/9)N - o(N) \qquad \text{(by Eq (3.5))}$$

Hence, the approximation ratio of the algorithm on this instance is at most $6/7 + o(1)$, bounded away from being a $(1-\epsilon)$ approximation for $\epsilon < 1/7$.

Although for randomly chosen subsets A and B, no lower bound on the size of H is known, we show in the following that if A and B are chosen *adversarially*, then there exist graphs for which solving Problem 1 requires storing a subgraph with *super linear* in n number of edges. Note that the number of queries of any non-adaptive algorithm is at least the number of edges in H and hence this bound on the number of edges in H implies that $\omega(n)$ queries are needed, or in other words, the number of per-vertex query needs to be a *function of* n. The existence of such a graph indicates a barrier to obtain a non-adaptive $(1-\epsilon)$-approximation algorithm.

To continue, we need a few definitions. For a graph $G(V, E)$, a matching M is called an *induced matching*, if there is no edge between the vertices matched in M, i.e., $V(M)$, except for the edges in M. A graph $G(V, E)$ is called an (r, t)-*Ruzsa-Szemerédi* graph ((r, t)-RS graph for short), if the edge set E can be partitioned into t *induced matchings* each of size r. Note than the number of edges in any (r, t)-RS graph is $r \cdot t$.

Suppose $G(L, R, E)$ is an (r, t)-RS graph with the parameter $r = N/3$ and induced matchings $M_1, \ldots, M_t$. For each $i \in [t]$, define A_i (resp. B_i) as $V(M_i) \cap L$ (resp. $V(M_i) \cap R$). Suppose we choose the pair (A, B) only from the set of pairs $\mathcal{F} := \{(A_1, B_1), \ldots, (A_t, B_t)\}$. Note that between any pair in $\mathcal{F}$, there is a matching of size $r = N/3$, and moreover all edges of G are partitioned between these matchings. If the subgraph $H(V, Q)$ has only $o(r \cdot t)$ edges, a simple counting argument suggests that for $1 - o(1)$ fraction of pairs in $\mathcal{F}$, only $o(r)$ edges between the pairs are present in H. Hence, H cannot be a solution to

Problem 1.

To complete the argument, we point out that there are (r, t)-RS graphs on $2N$ vertices with parameters $r = N/3$ and $t = N^{\Omega(1/\log\log N)}$ [48, 55]. These constructions certify that to solve Problem 1 when the sets A and B are chosen adversarially, one needs to store a subgraph with $n^{1+\Omega(1/\log\log n)} = \omega(n \cdot \text{polylog}(n))$ edges. In conclusion, while this result does not rule out the possibility of a non-adaptive $(1-\epsilon)$-approximation algorithm where the number of per-vertex queries is independent of n, it suggests that any such algorithm has to crucially overcome Problem 1 using the fact that the two sets A and B are chosen randomly instead of adversarially.

3.8. Conclusions and Future Work

In this chapter, we presented our study on the stochastic matching problem. We showed that there exists an adaptive $(1-\epsilon)$-approximation algorithm for this problem with $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ per-vertex queries and degree of adaptivity. We further presented a non-adaptive $(\frac{1}{2}-\epsilon)$-approximation algorithm with $O(\frac{\log(1/\epsilon p)}{\epsilon p})$ per-vertex queries. These results represent an exponential improvement over the previous best bounds of [25], answering an open problem in that work.

An interesting direction for future work is to design a non-adaptive algorithm that obtains a better than $\frac{1}{2}$-approximation while maintaining the property that the number of per-vertex queries is independent of n. Toward this direction, we highlighted a potential barrier to achieve a $(1-\epsilon)$-approximation non-adaptively and we believe that overcoming this barrier would play a key role in this line of research.

CHAPTER 4 : Convergence Time of Interdomain Routing

In the previous section, we discussed our work on the stochastic matching problem, which was focusing on data summarization. In this chapter, we will present our study on another graph problem, i.e., the convergence time of interdomain routing[1]. We first formalize the interdomain routing problem by introducing both the routing protocol, namely, the Border Gateway Protocol, and the notation of *partial preference systems*. We point out that convergence under partial preference systems and convergence under traditional (total) preference systems are connected through the concept of *path linearization*. Then, we consider rapid convergence for partial preference systems under different types of routing preferences and establishing dichotomy theorems. Finally, we study the problem of completing preferences while minimizing the path-length in the resulting stable routes. We will start with a brief introduction of the discovery of interdomain routing.

4.1. Background

The Internet is a network of networks, connected via a common routing protocol which allows data paths to be found that meet the local policy of each network. This protocol is 'BGP', the Border Gateway Protocol [97]. Its most important difference from other route-finding methods is the way in which local policy controls the selection and propagation of routes: rather than there being a single global definition of 'best' route (as in shortest-path protocols), every constituent network has its own interpretation. This is because of the asymmetry in the economic relationships among these network participants; accordingly, BGP is best perceived as a protocol that tries to compute a Nash equilibrium in a game where all parties are trying to obtain good routes (by their local definition) but are constrained by the choices of others (one cannot pick a route through a neighbor without agreement) [93, 46].

BGP policy configuration can be extremely complex and nuanced. Preferences among

[1]The full paper of this work can be found in [68].

possible network paths may be set arbitrarily. In practice, typical preferences are rather more structured. A standard configuration practice is for the first preference decision to be based on the identity of the neighbor from which the route was received; and then, among all routes coming from the most-favored neighbors, other characteristics such as path length are used to break ties [52, 27, 113]. Note that even in this restricted case, the preference classes over the neighbors may not be consistent across the entire network.

In general, network policies can conflict to the extent that BGP will be unable to find a stable solution: in this case, the protocol will *oscillate* indefinitely [81, 80, 88]. Furthermore, a set of policies may support the existence of multiple possible outcomes, which is typically also felt to be an error (because policy was not expressed well enough to yield the truly intended single outcome) [61]. There has been a great deal of work on identifying sufficient criteria for BGP to converge to a unique stable state [60, 62, 64, 107, 52, 104]. Unlike as previously suspected, the observed issues with convergence may not arise from router bugs or network anomalies, nor even from mistakes in the protocol definition, but may be *inherent* to the nature of the routing problem being solved. This was shown by the use of abstract models (in particular, *stable paths problems [63]*) displaying the same effects, without any of the complicating apparatus of BGP or its environment.

Even if BGP *does* converge, experience shows that it may take some time to do so [80, 47], and even in the absence of policy, when link failure occurs, the path-vector protocol used by BGP may still need exponential time to converge [75]. The slow convergence causes practical difficulties for network operators, and degradation or loss of service for their customers. The technology of 'route flap damping', which modifies the timing behavior of BGP in order to avoid propagating temporary oscillations, has been developed [108], criticized [87], and readjusted [94], but in a purely empirical fashion that does not address the root cause of delayed convergence.

The main focus in this paper is on the theoretical aspect of the convergence *time* of BGP. In particular, we aim to understand *when convergence is guaranteed*, how much time it

requires for the network to actually converge (or to stabilize). We study convergence time when different restrictions are applied on the structure of the preferences, and establish both necessary and sufficient conditions (i.e., dichotomy theorems) for convergence in *polynomial time*. To the best of our knowledge, prior to our work, the only related studies concerning polynomial time convergence are given by [52, 99], which only established polynomial time convergence for the Gao-Rexford criteria[2].

Our results are backed by a general model of routing preference, based on the idea of partial policy specification [67, 113]. While BGP requires all paths to be ranked in a linear order (as otherwise it cannot choose a single best path in all circumstances), operators do not actually think of policy in this way. It is more natural to imagine the linear preference order as being determined by a combination of a general operator-determined policy, and subsequent tie-breaking actions that do not reflect 'genuine' preferences. The general policy, as implemented in standard BGP systems via match-action rules, amounts to partitioning the set of possible paths into disjoint classes: within a class, no preference is given, and between the classes, preferences might exist. For example, 'all routes from neighbor 17' could be a class, which is preferred to the class 'all routes from neighbor 4'.

4.2. Our Results and Related Work

We first show that *convergence* is achieved when all possible routes can be put into a global linear order consistent with the given preferences (i.e., a linearization), which also matches known conditions for the existence of a unique stable solution [67]. We further establish that the construction of a linearization (when one exists), though the number of paths being linearized may be exponentially many, can always be done in time polynomial in the number of explicitly given preferences (Section 4.4).

For the setting where, instead of linearizing, the routers only follow the specified partial preferences and act indifferently between unordered paths (hence never voluntarily switch

[2] [99] also contains a general result regarding convergence time in terms of the number of *phases* (see Section 4.4.1 for the definition and more details).

to a path that is *not strictly better* than the current path), we present a detailed study of convergence time of networks (which are guaranteed to converge) with restricted preference systems. In particular, if each node only specifies preference over at most two paths, where each path has at most three hops, there still exist instances of networks that may take exponentially many (in the total number of nodes) steps before convergence. On the other hand, restricting the preference any further ensures poly-time convergence (Section 4.5). If a path's degree of preference is determined only by the identity of the next-hop neighbor, poly-time convergence is guaranteed [100]. However, using only the next two hops to determine preference already leads to instances that take exponentially many steps before convergence (Section 4.6).

Finally, we observe that a given partial policy may admit many possible linearizations which may result in distinct stable states for the network. A natural question is if one can efficiently compute a linearization that results in a stable state with some desirable properties. We consider the problem of computing a linearization that minimizes the hop-length of the longest path in the resulting stable state. We establish a strong hardness result by showing that the problem is NP-hard to approximate to within a factor of $\Omega(n)$. To complement the hardness result, we further provide a poly-time algorithm that finds a path linearization where the length of the longest path only incurs an additive error of at most l, where l is the length of the longest path in the preference.

Organization: This chapter is structured as follows. After introducing some fundamental concepts and notation (Section 4.3), we explore the linearization concept and its complexity (Section 4.4). We then show how possible restrictions of preference expressivity affects convergence time (Section 4.5, 4.6). Finally, we investigate the problem of completing policies that minimizes the length of the longest path (Section 4.7). We conclude (Section 4.8) by relating our results to prior work on the algorithmic complexity of BGP.

4.3. Preliminaries

In BGP, each network router (node) is capable of expressing independent preferences about its paths to each possible destination. It is well established that it suffices to focus on a single destination.

Definition 4.1 (Network)**.** *A* network N *is defined as a tuple* $\langle G(V,E), L, t\rangle$*, where* $G(V,E)$ *is a directed graph,* L *is the preference (or policies, see definition below), and* t *is the designated destination node.*

Throughout this paper, we denote by n the number of nodes in V. We assume that any $v \in V$ is connected to t in $G(V,E)$. Let $\mathcal{P}$ denote the set of all simple paths in $G(V,E)$ that terminate at t, and let $\mathcal{P}_v$ for each v in V, be the set of simple paths from v to t. The preference L contains path preferences expressed as partial orders on $\mathcal{P}_v$ for each node v in V.

Definition 4.2 (Preference)**.** *A preference* L_v *for a node* v *is a partial ordering* $\succ_v$ *on* all *simple paths from* v *to* t*,* P_v*. For any* $p, q \in P_v$*,* p *is 'better' than* q *iff* $p \succ_v q$*. Otherwise,* p *and* q *are unordered, and* v *is 'indifferent' between them.*

Generally, we write $p \succ_L q$ (or $p \succ q$ if L is clear from the context) if p is better than q for some v. In addition, we denote an empty path as ϵ, which is worse than any path in $\mathcal{P}$. We say a path p is *specified* in a preference L_v, denote by $p \in L_v$, if there exists a path q with $p \succ_v q$ or $q \succ_v p$.

In the protocol execution, each node tries to find a 'good' path to t, according to its own order, but subject to the requirement that it cannot choose a path unless the relevant neighbor has chosen the suffix of that path. Since forwarding is destination-based and hop-by-hop, a node may select a 'path', but data need not be constrained to follow that path if the other nodes along it have chosen differently. Therefore, it is more appropriate to say that a node chooses an outgoing edge to send traffic; it might continue to send traffic along that edge even after further network events have diverted the path. We now define the

possible routing states in the protocol execution more precisely.

Definition 4.3 (State)**.** *A* state *of the protocol execution is a function S from V to $E \cup \{\perp\}$, such that for each node v, other than t, we either have $S(v) = (v, u)$ when v has selected the edge (v, u) in E, or $S(v) = \perp$ if no neighbor is assigned to v.*

A node v is *connected* to t in a state S iff v is connected to t in the graph induced by the edges chosen in S. We will write $P_S(v)$ for the path induced by S from v to t. If v is not connected, then $P_S(v) = \epsilon$[3]. If v is connected, $P_S(v) = S(v)P_S(u)$ when $S(v) = (v, u)$ (this will always be a simple path); indeed, any intermediate node appearing on $P_S(v)$ is assigned the corresponding suffix of $P_S(v)$. When the network converges, these paths collectively form a tree directed towards t. During the execution, a node is constrained to only choose among the *available paths*.

Definition 4.4 (Available paths)**.** *If a node v has k out-neighbors $u_1, u_2, \ldots, u_k$ in $G(V, E)$, then the set of* available paths *for v in state S is* $\mathrm{AP}(S, v) = \{(v, u_i)P_S(u_i) \mid 1 \leq i \leq k, (v, u_i)P_S(u_i) \in \mathcal{P}_v\} \cup \{\epsilon\}$.

If the preference specifies a total order over all available paths, a node can always choose the best available path according to such an order. However, if only given a partial preference, for the unordered paths, a node still needs to make a decision among them. We consider the case where for each node v and paths starting from v, *those specified in L_v are better than the remaining*, and a node can change its state (and hence the state of the network) only if it has an *improving move*.

Definition 4.5 (Improvement and Stable State)**.** *A node v has an* improving *move in a state S iff there exists a path $p^* = (v, u^*)P_S(u^*)$ in $\mathrm{AP}(S, v)$ such that $S(v) \neq (v, u^*)$ and p^* is better than $P_S(v)$. The corresponding* improved state *S' will have $S'(v) = (v, u^*)$. A state is* stable *iff no node has an improving move.*

Consequently, whenever v is choosing among a set of paths without a total order, v can switch to an available path that is strictly better than his current path if any. If there is no

[3]Note that ϵ is an empty path while $\perp$ denotes an 'empty' edge.

such path, v will always stick with the current path (and hence has no improving move).

Remark 4.3.1 (The stable paths problem)**.** *Our partial preference system generalizes the case of the well known* stable paths problem *(SPP) [63] in the following sense. In SPP, each path is assigned with a* rank *(which reflects the priority) and for each node, paths with the same rank must have the same next-hop (so called the* strictness*). Since for each node, different paths with the same next-hop never appear simultaneously (the next-hop node can only have one path to t), order them arbitrarily will not affect the behavior of nodes in SPP. In other words, the preference of SPP for each node is (implicity) a total order over all paths. Partial preference system clearly captures total orders on paths, and it is more general since paths with different next-hop are allowed to be unordered.*

A node can only attempt to make an improvement when it is *activated* according to a *schedule*.

Definition 4.6 (Activation)**.** *An* activation *on $v \in V$ at a state S is a state transformation from S to a state S' such that only v can change its state to a improved state (if v has an improving move). We use ρ to denote the transformation function, i.e., $S' = \rho(S, v)$.*

Definition 4.7 (Schedule)**.** *A* schedule *is a sequence of activations defined by a function α from $\mathbb{N}$ to V, where $\alpha(\tau) = v$ means that the node v is activated at the time step τ. A schedule is* starvation-free *if for each node v and each time τ, there exists some $\Delta > 0$ such that $v = \alpha(\tau + \Delta)$. We will only consider starvation-free schedules.*

For an initial state S, we denote by $\mathsf{S}_\alpha^{(\tau)}$ the state reached after activating the list of nodes $\alpha(1), \alpha(2), \ldots, \alpha(\tau)$ in order (and $\mathsf{S}_\alpha^{(0)}$ is taken to be S). In other words, $\mathsf{S}_\alpha^{(\tau+1)} = \rho(\mathsf{S}_\alpha^{(\tau)}, \alpha(\tau+1))$, for any positive τ. The following claim is an immediate consequence of the protocol execution model.

Claim 4.3.2. *If $P_S(v) \neq \epsilon$ for some node v in some state S, for any schedule α and any $\tau > 0$, $P_{\mathsf{S}_\alpha^{(\tau)}}(v) \neq \epsilon$.*

Definition 4.8 (Convergence)**.** *A network has* converged *given a schedule α at time τ from a starting state S, if for any $\Delta > 0$, we have $\mathsf{S}_\alpha^{(\tau)} = \mathsf{S}_\alpha^{(\tau+\Delta)}$.*

Note that if a network converges, it always converges to a stable state.

Definition 4.9 (Convergence time)**.** *Given a starting state and a schedule, if τ is a time step at which the network N has converged ($\tau = +\infty$ if N never converges), we define the* convergence time *to be the number of improvements made up to τ. The* maximum convergence time*, denoted by $CT_{\max}(N)$, of N is the longest convergence time over all possible initial states and schedules.*

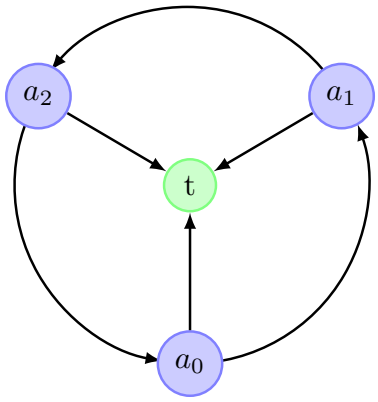
Note that since only the improving moves are counted, the convergence time is not necessarily equal to τ (though it is at most τ), and the definition of maximum convergence time is consistent for any τ at which N has converged.

4.4. Path Linearization

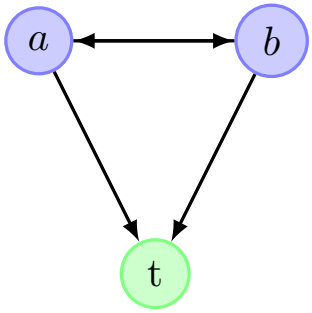
In this section, we show that whenever the specified partial preferences are free of dispute wheels (a particular cyclic arrangement of preferences), there is always a total order over all possible paths that is consistent with the preference such that a unique stable state will be reached after some bounded amount of time. We refer to such a network as a *linearizable network*, and to the process of finding such a total order as *path linearization*. Our approach is based on exploiting the structure of the "path digraph" associated with the input instance, taking advantage of a well-known connection between path digraphs and dispute-wheels. Moreover, we provide a path linearization algorithm that, though computing a total order on possibly exponentially many paths, runs in time polynomial in the size of the input graph and the preference.

4.4.1. Path Digraph and Dispute Wheels

A *path digraph* is a graph representation for 'BGP-like' path problems, where individual nodes' preferences govern route selection in a path-vector algorithm. For a network $\langle G(V, E), L, t\rangle$, the nodes of the path digraph are all the simple paths in G. The nodes for paths p and q are connected by a directed edge (p, q) if either p is a suffix of q or some node in G prefers p to q. Thus in the path digraph any suffix p of a path q (denoted by $p \rightarrow q$)



(a) Topology of an unstable network.



(b) Topology of an network with two stable states.

Figure 4: Some network examples.

is always considered better than the path q itself.

Definition 4.10 (Linearization)**.** *For a given network $\langle G(V,E), L, t\rangle$, a* linearization $\succ_{\mathcal{P}}$ *of all paths $\mathcal{P}$ is a total order compatible with both L and the suffix relations. Specifically, for any $p, q \in \mathcal{P}$, (a) if $p \succ_L q$, then $p \succ_{\mathcal{P}} q$ (preference compatibility), and (b) if $p \to q$ then $p \succ_{\mathcal{P}} q$ (suffix compatibility).*

Define a *phase* of a schedule to be an interval of time during which all nodes are activated at least once. Then, linearizable networks have the following property.

Theorem 4.1. *Any linearizable network will converge to a unique stable state. Moreover, n phases suffice for any linearizable network on n nodes to converge under any initial state and any schedule, where n is the number of nodes.*

Proof. By [63, 99], absence of dispute wheels implies that the network always converges to a unique stable state, and convergence will happen in at most n phases. Absence of dispute wheels is equivalent to acyclicity of the path digraph [67], which is equivalent to linearizablility of a network. □

On the other hand, without such a global linearization, convergence is not guaranteed. Figure 4a shows a typical example where the network will never converge. Assume each node a_i has preference $a_i a_{i+1} t \succ a_i t \succ a_i a_{i+1} a_{i+2} t$, where subscripts are interpreted modulo

3. In fact, this network has no stable state, and hence it can never converge. To see this, consider the number of nodes choosing the direct edge to t in a stable state if there exists one. If none of the nodes takes the path $a_i t$, any a_i will take $a_i t$ when activated (hence unstable). If only a_0 chooses the direct edge $a_0 t$, a_2 will choose $a_2 a_0 t$ and leave a_1 no choice but also choosing the direct edge to t. If a_0, a_1 both choose the direct edges to t, then a_0 will switch to $a_0 a_1 t$. If all nodes choose the direct edge to t, then a_0 will switch to $a_0 a_1 t$.

4.4.2. A Polynomial Time Algorithm for Linearization

We assume in the following that the input network is linearizable, and show that an efficiently computable linearization always exists. As our goal is to create a linearization for possibly exponentially many paths in polynomial time, the output can not be an explicit representation of the linear order. Hence we design a poly-time computable function that takes as input an ordered pair of simple paths (p, p') and outputs *yes* whenever p is ranked higher than p' in the linearization, and *no* otherwise. We establish the following.

Theorem 4.2. *Any linearizable network has a linearization that can be computed in polynomial time.*

To linearize a network $\langle G(V, E), L, t\rangle$, we first consider the path digraph with respect to only the *specified paths* $\mathcal{P}_L$. Formally, each path in $\mathcal{P}_L$ is (i) the trivial path that has only one node t, (ii) a path in the preference system L, or (iii) a subpath (i.e., a suffix) of some path in L. We call the following linear order of $\mathcal{P}_L$ a *spine*.

Definition 4.11 (Spine)**.** *A spine on $\langle G(V, E), L, t\rangle$ is a linear order $\succ_{\mathcal{P}_L}$ on $\mathcal{P}_L$ such that for any $p, q \in \mathcal{P}_L$, (a) if $p \succ_L q$ then $p \succ_{\mathcal{P}_L} q$, and (b) if $p \to q$ then $p \succ_{\mathcal{P}_L} q$. Note that path t must be the largest element in any spine.*

A spine $\succ_{\mathcal{P}_L}$ can be created efficiently since the total number of paths in $\mathcal{P}_L$ is at most $|L| + n|L| + 1 = O(n|L|)$ (which respectively corresponds to the paths in L, the suffixes of paths in L, and the trivial path t) and a topologically sorted order of $\mathcal{P}_L$ can found in $O(n^2|L|^2)$ time.

Now, we show how to use a spine $\succ_{\mathcal{P}_L}$ to find a linearization of the network. For each path p in $\mathcal{P} \setminus \mathcal{P}_L$, among all suffixes of p that are in $\mathcal{P}_L$, map p to the smallest suffix according to the order $\succ_{\mathcal{P}_L}$; the order between two paths p, q that are mapped to two different paths p', q' in $\mathcal{P}_L$ is compatible with the order between p' and q' in $\succ_{\mathcal{P}_L}$; all paths assigned to the same path in $\mathcal{P}_L$ are ordered lexicographically.

Claim 4.4.1. *If a network $\langle G(V, E), L, t\rangle$ is linearizable, there must exist a spine on $\mathcal{P}_L$.*

In fact, the restriction of any linearzation to $\mathcal{P}_L$ is a spine, by Definitions 4.10 and 4.11.

Definition 4.12 (Vertebra)**.** *For any path p in $\mathcal{P} \setminus \mathcal{P}_L$, the* vertebra *of p is $v(p) = \min_{\succ_{\mathcal{P}_L}} \{q \in \mathcal{P}_L \mid q \to p\}$, i.e., the minimal spine path among all of its suffixes. Note that because t is in $\mathcal{P}_L$, $v(p)$ is always well defined.*

We now prove Theorem 4.2 by demonstrating the following order.

Definition 4.13 (Spinal order)**.** *Given a spine $\succ_{\mathcal{P}_L}$ on network $\langle G(V, E), L, t\rangle$, define the* spinal order *$\succ_{\mathcal{P}}$ on $\mathcal{P}$ by $p \succ_{\mathcal{P}} q$ if and only if $v(p) \succ_{\mathcal{P}_L} v(q)$, or $v(p) = v(q)$ and $p \geq_{\text{lex}} q$.*

We introduce the following three lemmas to approach Theorem 4.2.

Lemma 4.4.2. *The spinal order is a linearization of $\mathcal{P}$.*

Proof. We must show that the order $\succ_{\mathcal{P}}$ is a total order that is consistent with preference and suffix compatibility rules.

First of all, the order $\succ_{\mathcal{P}}$ is indeed a total order.

- **Reflexivity.** Obvious from the definition.
- **Transitivity.** Suppose $p \succ_{\mathcal{P}} q \succ_{\mathcal{P}} r$; consider $v(p)$, $v(q)$ and $v(r)$. If $v(p) = v(q) = v(r)$, we must have $p \geq_{\text{lex}} q \geq_{\text{lex}} r$ and so $p \geq_{\text{lex}} r$. Hence $p \succ_{\mathcal{P}} r$. If $v(p) \succ_{\mathcal{P}_L} v(q)$, then either $v(q) \succ_{\mathcal{P}_L} v(r)$ or $v(q) = v(r)$. For the first case, since the spine is transitive, we have $v(p) \succ_{\mathcal{P}_L} v(r)$; in the second case, $v(q) \succ_{\mathcal{P}_L} v(r)$ trivially. Thus $v(p) \succ_{\mathcal{P}} v(r)$. The argument for $v(q) \succ_{\mathcal{P}_L} v(r)$ is exactly parallel.

- **Antisymmetry.** Suppose $p \succ_{\mathcal{P}} q$ and $q \succ_{\mathcal{P}} p$. Clearly we must have $v(p) = v(q)$, from which it follows that $p \geq_{\text{lex}} q$ and $q \geq_{\text{lex}} p$. Hence $p = q$ as required.

- **Totality.** Recall that v is defined for all paths. Suppose that $\neg(p \succ_{\mathcal{P}} q)$, for some distinct p and q. Then $\neg v(p) \succ_{\mathcal{P}_L} v(q)$, and since the spine is a total order, we have either $v(q) \succ_{\mathcal{P}_L} v(p)$ or $v(p) = v(q)$. The first case directly implies $q \succ_{\mathcal{P}} p$. In the second, it must be that $\neg(p \geq_{\text{lex}} q)$ according to the definition of this order. Since $\geq_{\text{lex}}$ is a total order, we must have $q \geq_{\text{lex}} p$, from which it follows that $q \succ_{\mathcal{P}} p$, as required.

Secondly, the order is compatible with the preferences in L. Suppose that $p \succ_L q$. Then $p = v(p)$, $q = v(q)$ and $v(p) \succ_{\mathcal{P}_L} v(q)$, so $p \succ_{\mathcal{P}} q$ as required.

Finally, the order is compatible with the suffix relation. Suppose that $p \to q$. By definition of vertebra, it must be that $v(p)$ is at least as good as $v(q)$ on the spine, because any suffix of p is also a suffix of q. If $v(p) \succ_{\mathcal{P}_L} v(q)$ then $p \succ_{\mathcal{P}} q$ and we are done. Otherwise, $v(p) = v(q)$; but in this case, p is a shorter path than q and so lexicographically precedes it. We have $p \geq_{\text{lex}} q$ and $p \succ_{\mathcal{P}} q$ as required. □

Lemma 4.4.3. *If a spine exists, then it can be constructed in $O(|\mathcal{P}_L|^2)$ time.*

Proof. Recall that a spine is a linear order on $\mathcal{P}_L$, which extends both the preferences in L and the suffix relations. Let Π be a binary relation on $\mathcal{P}_L$ which contains exactly those pairs of paths (p, q) for which $p \succ_L q$ or $p \to q$. A topological sort of Π will either fail, indicating the presence of a cycle and hence nonexistence of a spine, or succeed, and produce a spine.

Extending L to Π can be done in $O(|\mathcal{P}_L|^2)$ time. A topological sort of all paths in Π can be done in $O(|\mathcal{P}_L| + |\mathcal{P}_L|^2)$ time, where $|\mathcal{P}_L|^2$ is an upper bound on the number of relations in Π. Therefore, the entire running time is bounded by $O(|\mathcal{P}_L|^2)$. □

Lemma 4.4.4. *Given any two paths p and q from $\mathcal{P}$, the determination of whether $p \succ_{\mathcal{P}} q$ or $q \succ_{\mathcal{P}} p$ holds can be made in polynomial time.*

Proof. We may assume a spine has already been constructed, since this takes polynomial time in any case. To identify the ordering of p and q, we must evaluate and compare $v(p)$ and $v(q)$, and if they are equal, then compare p and q lexicographically.

We first show that it takes at most $O(n)$ time to evaluate $v(p)$ for a given p. This is achieved simply by enumerating all suffix paths of p starting from longest to shortest, and output the first path that is in $\mathcal{P}_L$.

Comparison of $v(p)$ and $v(q)$ is done according to the spine. If they are equal, lexicographic comparison of p and q takes $O(n)$ time.

We conclude that preference queries, for two given paths, take $O(n + |\mathcal{P}_L|)$ time overall, once a spine has been constructed. □

Proof of Theorem 4.2. Lemma 4.4.1 guarantees that there exists at least one spine. By Lemma 4.4.2, the corresponding spine order is a linearization of $\mathcal{P}$. In addition, by Lemma 4.4.3 and 4.4.4, we can construct as well as determine preferences between two paths in polynomial time. □

4.5. Convergence Time for Restricted Preferences: a Dichotomy Theorem

Although path linearization guarantees convergence after n phases of execution, this does not mean that the convergence time is polynomially bounded since there could be arbitrarily long sequences of improving moves during a phase. In this and the next sections, we study the convergence time of linearizable, but not linearized, networks (i.e., executing the protocol under partial preferences) over restricted families of preference systems.[4] We start with the following families of preference systems.

Definition 4.14 ($\langle s, l\rangle$-Preference Systems)**.** *A preference L is a $\langle s, l\rangle$-preference system iff for the preference L_v of each node v, (i) there are at most s paths in $|L_v|$, i.e., $|L_v| \leq s$,*

[4]The execution model for partial preferences implicitly assumes that paths specified in the preference is better than the rest. By simply examining the resulting path digraph, it can be verified that having these additional preferences does not affect the linearizability of a network.

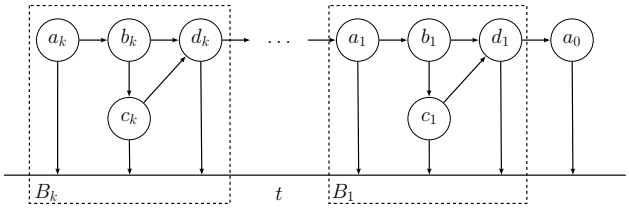


Figure 5: A network with constant preference size and constant-length paths, that takes an exponentially long time to converge.

and (ii) for each path p in L_v, the length of p is at most l, i.e., $\max_{p \in L_v} \{|p|\} \leq l$.

We establish here the following dichotomy: even a linearizable network with only a $\langle 2, 3 \rangle$-preference system could encounter exponentially many improving moves before convergence, while any linearizable network with a $\langle 2, 2 \rangle$-preference system (resp. a $\langle 1, 3 \rangle$-preference system) always converges after at most n^2 (resp. $2n$) improvements.

4.5.1. Exponential Convergence Time for $\langle 2, 3 \rangle$-Preferences

Our first result shows that there exists a family of (even acyclic) networks such that the preference of any node contains at most two paths where each path has length at most three, and yet convergence may take time exponential in the size of the network.

Theorem 4.3. *For any $k \geq 1$, there exists a network $N\langle G(V, E), L, t \rangle$, where G is a DAG on $n = 4k + 2$ nodes, and L is a $\langle 2, 3 \rangle$-preference system, s.t. the maximum convergence time of N is $2^{\Omega(n)}$.*

Note that G is a DAG immediately implies that the network is free of dispute wheel, and hence linearizable. A network N satisfying the conditions of the above theorem is depicted in Figure 5, with the destination node t shown as the 'ground'. The network consists of a special node a_0, followed by $k = \lfloor n/4 \rfloor$ blocks B_i of four nodes each, named a_i, b_i, c_i and d_i. The preferences for each node in a block B_i are $L_{a_i} = \{a_i b_i d_i t \succ a_i t\}$, $L_{b_i} = \{b_i c_i t \succ b_i d_i t\}$, $L_{c_i} = \{c_i d_i t \succ c_i t\}$ and $L_{d_i} = \{d_i a_{i-1} t \succ d_i t\}$. The number of paths in the preference of for each node is 2 and every path has at most 3 hops.

The central idea is a pattern of activations for the nodes in each block B_i, whereby a 'flip' for node a_{i-1} (that is, when the node changes its state and then returns to the previous

state) triggers two flips for node a_i. Accordingly, the sequence of blocks can be made to amplify a single flip at a_1 into exponentially many flips for the subsequent a_i.

For the interest of space, in the following, we will present the activation sequence of each block and explain how it leads to amplification, while a detailed proof of correctness is supplied in [68]. The order of each block is defined as follows.

Definition 4.15 (Block Order)**.** *For $1 \leq i \leq k$, let σ_i be the following ordering of nodes in B_i:*

$$d_i, a_i, b_i, a_i, c_i, b_i, a_i, d_i, c_i, a_i.$$

We will refer to this ordering as the block order *of B_i.*

We consider the block order of activation starting with the following state.

Definition 4.16 (State S_1)**.** *For any $1 \leq i \leq k$, the S_1 state of the block B_i is defined as $S_1(a_i) = (a_i, t)$, $S_1(b_i) = (b_i, d_i)$, $S_1(c_i) = (c_i, t)$, and $S_1(d_i) = (d_i, a_{i-1})$.*

The activation of the block order on B_i starting from the state S_1 is illustrated in Figure 6, where the red (or thick) edges represent the current state. The following lemma provides the essential 'amplification' step for the exponential time result. Recall that we say that a node 'flips' when it changes its assigned edge and then changes back. The lemma shows that a flip for node a_{i-1} can cause a double flip for node a_i by activating the nodes in B_i according to the block order (with the activation of a_{i-1} interposed). Therefore the node a_{i+1} can be made to flip four times, and so on.

Lemma 4.5.1. *For any $1 < i \leq k$, suppose that the block B_i is in state S_1, the state of a_{i-1} is (a_{i-1}, b_{i-1}), and the node a_{i-1} will shortly change to (a_{i-1}, t). Then there exists a schedule such that each node makes an improving move when activated, and the final state of B_i is still S_1. Moreover, during activating under this schedule, the state of d_i flips once (from (d_i, a_{i-1}) to (d_i, t) and then back), and a_i flips twice (from (a_i, t) to (a_i, b_i) and back, twice).*

Proof. The schedule we use will activate the nodes of B_i in the block order σ_i with the

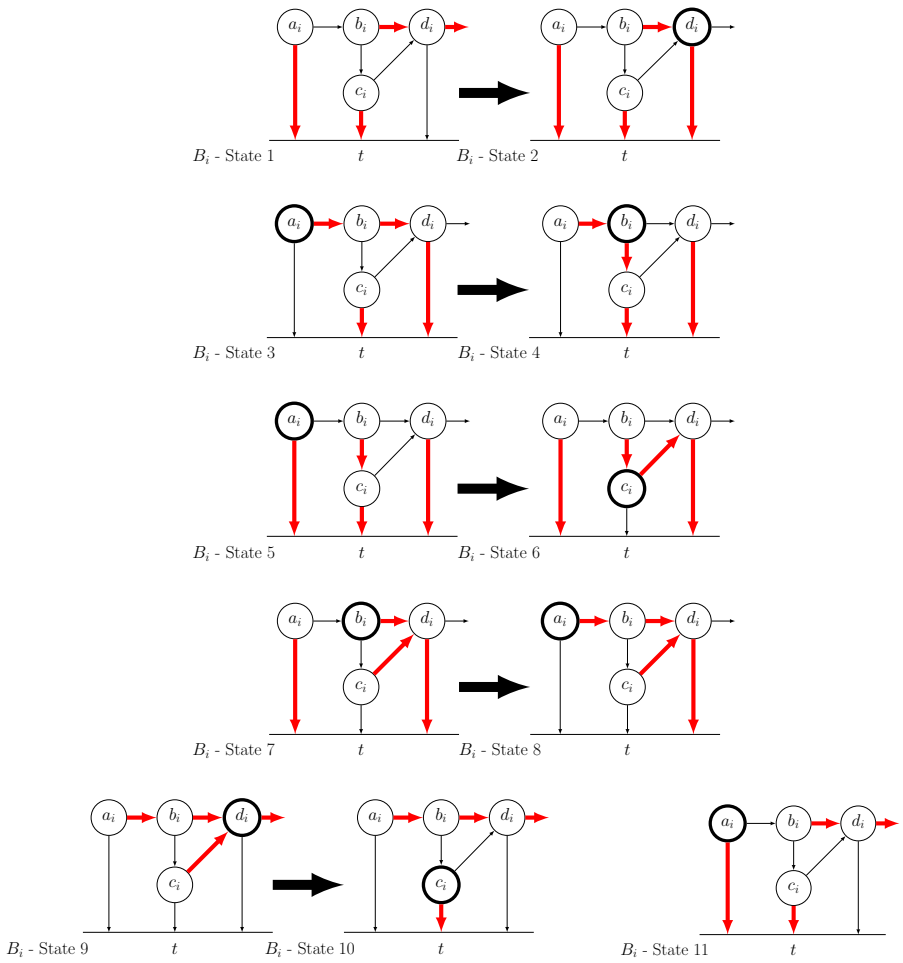


Figure 6: States resulting from applying activation sequence σ_i to block B_i.

activation of a_{i-1} to adopt (a_{i-1}, t) occurring part way through.

The progression of states leading to required effect is shown in Figure 6, and explained in detail in Table 1. Note that a_{i-1} will activate and adopt (a_{i-1}, t) after state 2 but before state 9.

Consequently, the transition to state 2, where node d_i switches from an unranked path beginning with (d_i, a_{i-1}) to its second-best path $d_i t$ is an improving move, and so is the transition to state 9, where a_{i-1} has switched to (a_{i-1}, t) and so d_i can improve to its best path, $d_i a_{i-1} t$.

The claims of the lemma statement can readily be verified from the presented sequence. □

State	Activate	Old edge	New edge	Improvement reason
1	d_i	(d_i, a_{i-1})	(d_i, t)	$d_i t \succ_{d_i} d_i a_{i-1} b_{i-1} \dots$
2	a_i	(a_i, t)	(a_i, b_i)	$a_i b_i d_i t \succ_{a_i} a_i t$
3	b_i	(b_i, d_i)	(b_i, c_i)	$b_i c_i t \succ_{b_i} b_i d_i t$
4	a_i	(a_i, b_i)	(a_i, t)	$a_i t \succ_{a_i} a_i b_i c_i t$
5	c_i	(c_i, t)	(c_i, d_i)	$c_i d_i t \succ_{c_i} c_i t$
6	b_i	(b_i, c_i)	(b_i, d_i)	$b_i d_i t \succ_{b_i} b_i c_i d_i t$
7	a_i	(a_i, t)	(a_i, b_i)	$a_i b_i d_i t \succ_{a_i} a_i t$
8	d_i	(d_i, t)	(d_i, a_{i-1})	$d_i a_{i-1} t \succ_{d_i} d_i t$
9	c_i	(c_i, d_i)	(c_i, t)	$c_i t \succ_{c_i} c_i d_i a_{i-1} \dots$
10	a_i	(a_i, b_i)	(a_i, t)	$a_t \succ_{a_i} a_i b_i d_i a_{i-1} \dots$

Table 1: Sequence of improving moves for Lemma 4.5.1.

4.5.2. An Upper Bound on Convergence Time

We now investigate the maximum convergence time of linearizable networks. Firstly, note that any $\langle 1, l\rangle$-preference system for any positive integer l, will converge after at most $2n$ improvements since a node v can only switch from being not connected to t to choosing some neighbor u where u has a path to t, or switch from choosing some neighbor u to choosing the neighbor of the only path in v's preference (when this path becomes available). Hence, we only need to establish maximum convergence time for $\langle 2, 2\rangle$-preference systems, and combining with our construction for showing that $\langle 2, 3\rangle$-preference systems lead to exponentially many improvements (Theorem 4.3), we have a dichotomy theorem. In fact, we will establish the maximum convergence time for networks with $\langle 2, l\rangle$-preference systems for any positive integer l, which, as a special case, implies poly-time convergence for $\langle 2, 2\rangle$-preference systems.

We define the function $\mathcal{L}(l, n)$ for any positive integers l and n, to be the sum of a list of integers $\mathcal{L}(l, n) = \sum_{i=1}^{n} a_i$, where $a_1 = 0$ and $a_i = (l-1)a_{i-1} + 2$ for all i, and establish the following theorem.

Theorem 4.4. *For any positive integer l, for any linearizable network $N\langle G(V, E), L, t\rangle$ where L is a $\langle 2, l\rangle$-preference system, the maximum convergence time of N is at most $\mathcal{L}(l, n)$, where n is the number of nodes in G.*

We first establish that the total number of improvements made by any node v is upper bounded by the total number of improvements made by the nodes on the best path of v (Lemma 4.5.2). To utilize this fact, we introduce a process for finding a sequence of the nodes such that every node v appears after any other node on the best path of v (Lemma 4.5.3). Examining the nodes under this sequence, we show that for any i, the i-th node in the sequence can make at most a_i improvements, hence proving that $\mathcal{L}(l, n)$ is an upper bound of the total number of improvements.

We denote by $Imp(v)$ the maximum number of improvements that the node v can make. (The proofs of Lemma 4.5.2, 4.5.3 are deferred to [68].)

Lemma 4.5.2. *For any node v in a $\langle 2, l\rangle$-preference system network whose best path is denoted by $vu_1u_2 \dots u_jt$, $Imp(v) \leq \sum_{i=1}^{j} Imp(u_i) + 2$.*

We use the notion of "good" node to define/find a sequence of the nodes when examining their number of improvements.

Lemma 4.5.3. *For any linearizable network $\langle G(V, E), L, t\rangle$, for any $T \subsetneq V$ with $t \in T$, there exists at least one node $v \in V \backslash T$ s.t. either $L_v = \emptyset$, or for the best path of v $vu_1 \dots u_jt$, $u_i \in T$ for all i. We call such a node v a* good *node for T.*

Proof of Theorem 4.4. We will create a list of subsets of V, $\{A_i\}_{i=1}^{n}$ by starting from $A_1 = \{t\}$, adding one node every step, and ending with $A_n = V$. Let $a_i = \max\{Imp(v) \mid v \in A_i\}$. We will prove in the following that $a_i \leq (l-1)a_{i-1} + 2$. Since the node added at the i-th step can make at most a_i improvements, by summing up the number of improvements of all nodes, we achieve the upper bound of $\mathcal{L}(l, n)$ total improvements.

Starting from $A_1 = \{t\}$, we now show the transition from A_{i-1} to A_i, i.e., how to pick a node v that forms $A_i = A_{i-1} \cup \{v\}$ with $a_i \leq (l-1)a_{i-1}+2$. By Lemma 4.5.3 with $T = A_{i-1}$, there exists a node v with either empty preference, or for v's best path, $vu_1u_2 \dots u_jt$, all nodes except v belong to A_{i-1}. We pick any such node v (i.e., any good node for A_{i-1}). For the first case, v has an empty preference implies that the only improvement v can make is from

$\perp$ to one of its neighbor. For the second case, by Lemma 4.5.2, $Imp(v) \leq \sum_{i=1}^{j} Imp(u_i)+2$. Since $j \leq (l-1)$ and all u_i belongs to A_{i-1}, we have $Imp(v) \leq \sum_{i=1}^{j} a_{i-1}+2 \leq (l-1)a_{i-1}+2$. Since $a_i = \max\{a_{i-1}, Imp(v)\}$, $a_i \leq (l-1)a_{i-1}+2$. □

A simple calculation shows that $\mathcal{L}(2,n) \leq n^2$, and for any $l \geq 3$, $\mathcal{L}(l,n) \leq 2(l-1)^n$. Therefore, in particular, for any network with a $\langle 2,2\rangle$-preference system, the maximum convergence time is at most n^2. For a network with $\langle 2,3\rangle$-preference system, the maximum convergence time is at most 2^{n+1}. Combined with Theorem 4.3, which shows that there exist networks with $\langle 2,3\rangle$-preferences that take $2^{\Omega(n)}$ time to converge, we establish that the *convergence time complexity* of networks with $\langle 2,3\rangle$-preference systems is $2^{\Theta(n)}$.

4.6. Hop-Based Preference Systems

In this section we examine the maximum convergence time of hop-based preference systems.

Definition 4.17 (k-Hop Preference Systems)**.** *In a* k-hop preference system*, every node chooses paths based only on the next* k *hops of the paths.*

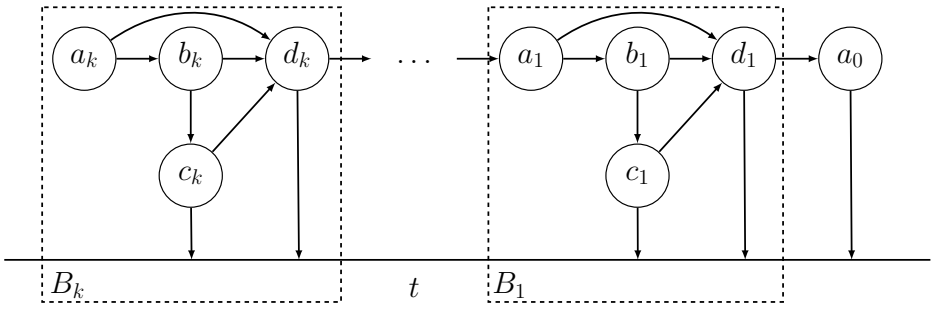
We use $v\langle u\rangle k* \succ_v v\langle w\rangle k*$ to denote that the prefix $v\langle u\rangle k$ is better than $v\langle w\rangle k$ for the node v. The well-known preference scheme of Gao and Rexford [52] is an example of a 2-hop preference system. All adjacencies are classified as customer-provider or peer-peer. Nodes are required to prefer customer routes over all others, which is a 1-hop preference rule. In addition, so-called 'valley' paths are worst of all: a path that go from a provider, 'down' to a customer, and then back 'up' to another provider. The extended transit provider guidelines of Liao et al. [83] can be treated in the same way. However, it is clear that the definition of k-hop preference is more general than these, even for the case when k is 2. The simplest case, when k is 1, covers the pure local preference scheme where initial preference decisions are based on the identity of the next-hop neighbor.

We establish here the following dichotomy result: any network with a 1-hop preference system converges in linear time while there exists a family of linearizable networks with 2-hop preferences systems such that the maximum convergence time is exponential.

4.6.1. Exponential Convergence Time for 2-Hop Preferences

We start by establishing the exponential convergence time result for linearizable networks with 2-hop preference systems. Note that the example shown in Figure 4a can be directly transformed into a 2-hop preference system which suggests that there are networks with 2-hop preference systems that never converge, which of course have unbounded maximum convergence time. The focus of this section is to show exponential convergence time for networks guaranteed to converge (in fact, even for DAGs).

Theorem 4.5. *For any $k \geq 1$, there exists a network $N\langle G(V,E), L, t\rangle$ with $n = 4k+2$ nodes forming a DAG and a 2-hop preference system such that the maximum convergence time of N is $2^{\Omega(n)}$.*



(a) Topology.

Node	1	2	3	4
a_i	$d_i a_{i-1}*$	$b_i d_i*$	$d_i t$	$b_i c_i*$
b_i	$c_i t$	$d_i t$	$c_i d_i*$	$d_i a_{i-1}*$
c_i	$d_i t$	t	$d_i a_{i-1}*$	
d_i	$a_{i-1} d_{i-1}*$	t	$a_{i-1} b_{i-1}*$	

(b) Preferences from best (1) to worst (4).

Figure 7: A network with 2-hop preferences and exponential convergence time.

A slight modification of the construction in Theorem 4.3 will demonstrate the same result for 2-hop preference systems. Consider the network whose topology is shown in Figure 7a. The only difference from the network used in Theorem 4.3 is that here a_i has an edge pointing to d_i, instead of t. The 2-hop preferences are shown in Table 7b.

We show the activation sequence that requires exponentially many steps for convergence on

this network in the following. The proof of Theorem 4.5 is omitted since it is very similar to Theorem 4.3.

As before, we break the network into blocks and denote them as B_i and define a new S_1 state similar to the previous case.

Definition 4.18 (S_1 state, 2-hop case)**.** *The state S_1 of B_i is the state $S_1(a_i) = (a_i, d_i)$, $S_1(b_i) = (b_i, d_i)$, $S_1(c_i) = (c_i, t)$, and $S_1(d_i) = (d_i, a_{i-1})$.*

Lemma 4.6.1. *For any B_i, starting from state S_1, there is an activation sequence of improving moves, terminating in S_1 again, during which node d_i changes from (d_i, a_{i-1}) to (d_i, t) and back, and node a_i changes from (a_i, d_i) to (a_i, b_i) and back, twice.*

State	a_i	b_i	c_i	d_i
1	(a_i, d_i)	(b_i, d_i)	(c_i, t)	(d_i, a_{i-1})
2	(a_i, d_i)	(b_i, d_i)	(c_i, t)	$\star(d_i, t)$
3	$\star(a_i, b_i)$	(b_i, d_i)	(c_i, t)	(d_i, t)
4	(a_i, b_i)	$\star(b_i, c_i)$	(c_i, t)	(d_i, t)
5	$\star(a_i, d_i)$	(b_i, c_i)	(c_i, t)	(d_i, t)
6	(a_i, d_i)	(b_i, c_i)	$\star(c_i, d_i)$	(d_i, t)
7	(a_i, d_i)	$\star(b_i, d_i)$	(c_i, d_i)	(d_i, t)
8	$\star(a_i, b_i)$	(b_i, d_i)	(c_i, d_i)	(d_i, t)
9	(a_i, b_i)	(b_i, d_i)	(c_i, d_i)	$\star(d_i, a_{i-1})$
10	(a_i, b_i)	(b_i, d_i)	$\star(c_i, t)$	(d_i, a_{i-1})
11	$\star(a_i, d_i)$	(b_i, d_i)	(c_i, t)	(d_i, a_{i-1})

Table 2: State sequence of block B_i.

We use the same block order as before (Definition 4.15). Table 2 shows the resulting states; the annotation '$\star$' shows which node made the improving move.

When a_i changes to (a_i, b_i), d_{i+1} will lose its best path and switch to the second best (d_i, t), which is always available. When a_i flips back to (a_i, d_i), the best path of d_{i+1} appears again, and d_{i+1} will change to it. As a result, each flip of d_i will cause a_i to flip twice and each flip of a_i will cause d_{i+1} to flip twice. The construction proceeds as in Theorem 4.3.

We note that it is straightforward to extend our construction for 2-hop system to ℓ-hop, for any $\ell > 2$. As long as all ℓ-hop prefixes with the same first 2 hops are adjacent to each other,

the extended preference has the exact same effect as the original. Thus the construction given in Theorem 4.5 in fact shows potential exponential-time convergence for completely ordered ℓ-hop preference systems for any $\ell \geq 2$.

4.6.2. Linear-Time Convergence for 1-Hop Preferences

It is well known that whenever BGP preferences constitute a 1-hop preference system, the network always converges to a stable state after a linear number of improvements.

Theorem 4.6. *For any network $N\langle G(V,E), L, t\rangle$, where L is a 1-hop preference system, the maximum convergence time of N is at most $n+m$.*

Proof. We prove this through a simple potential function argument. For any state S of the network, we define an n-dimensional vector W that records the 'quality' of the current solution for each node. Specifically, for each node $v \in V$, we set entry $W(v) = i$ if $S(v)$ is the i-th best next hop for v, and $W(v) = |k_v| + 1$ otherwise (i.e. if $S(v) = \perp$); here k_v denotes the number of out-neighbors of v.

By Claim 4.3.2, after making an improvement, v will be connected to t and can always stick with its current path, and hence the value $W(v)$ is non-increasing over time.

On the other hand, whenever v makes an improvement, it must switch to a better neighbor, and the value of $W(v)$ will strictly decrease. It follows that the total number of improving moves is bounded by

$$\sum_{v \in V} W(v) \leq \sum_{v \in V} (k_v + 1) \leq m + n$$

Hence the system always converges to a stable state in $n+m$ steps. □

Note that Theorem 4.6 does not rely on absence of dispute wheels. This indicates that for 1-hop preference, there always exists at least one stable state. The network in Figure 4b is a typical example where there is a dispute wheel (which leads to multiple stable states) but the network always converges. Here the nodes a and b prefer each other to the direct edge

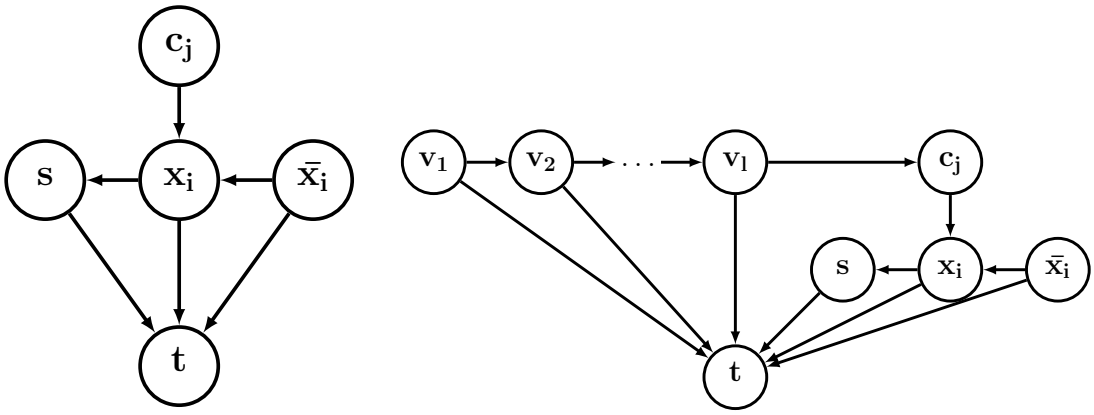
to t. There are two stable states: $\{(a,b),(b,t)\}$ and $\{(b,a),(a,t)\}$. Which one is reached depends on which node is activated first.

Note that if allowing activation for multiple nodes simultaneously for networks with dispute wheels, even if there are stable states, convergence is not guaranteed. Again, consider the network in Figure 4b. If we start from the state $S(a) = (a,t)$ and $S(b) = (b,t)$, and activate both a and b. The path abt is available to a, and the path bat is available to b. Hence the new state will be $S(a) = (a,b)$ and $S(b) = (b,a)$. If we activate both a and b again, both of them will realize that they are not connected to t, and hence will switch back to $S(a) = (a,t)$ and $S(b) = (b,t)$. Therefore, the network will keep flipping between those two states and never converge.

4.7. Linearization that Minimizes Path-Length

In this section, we study the problem of finding a path linearization that minimizes the length of the longest path in the stable state, which we refer to as the path-length minimization problem (PLM). We define the "length" of a path linearization to be the length of the longest path in the stable state, and hence PLM is to find a path linearization with the minimum length. We will only linearize the paths to the extent where the stable state is unique, and the remaining paths can either be linearized arbitrarily, or left unordered.

In the absence of preferences, PLM can be solved by simply performing a breath first search from the sink. Surprisingly, the presence of preference makes the problem almost impossible to tackle: it is NP-hard to approximate PLM to within a factor of $\Omega(n)$, (n is the number of nodes in the network). We further complement the hardness result by presenting an algorithm that finds a path linearization with length at most $(OPT + l)$, where OPT is the minimum length achievable by a path linearization and l is the length of the longest path in the preference.



(a) Basic gadget for the clause $c_j = x_i$.

(b) Final gadget for the clause $c_j = x_i$.

Figure 8: The construction for showing the hardness of PLM.

4.7.1. Hardness of the Path-Length Minimization Problem

Theorem 4.7. *For any $2 \leq l \leq \Theta(n)$, it is NP-hard to distinguish whether the minimum length of a path linearization for a given network on n nodes is 2 or at least l.*

Proof. We use a reduction from 3SAT. Given a 3SAT instance $C = c_1 \wedge c_2 \wedge \ldots c_y$ with variables $x_1, x_2, \ldots, x_z$, we create the sink t and a special node s with edge (s, t). For each variable x_i, create two nodes that respectively represent x_i and $\overline{x_i}$, with edges (x_i, t), (x_i, s), $(\overline{x_i}, t)$, and $(\overline{x_i}, x_i)$. For each clause c_j, create a node that represents c_j with edges (c_j, x_i) if x_i is in c_j, and $(c_j, \overline{x_i})$ if $\overline{x_i}$ is in c_j (see Figure 8a for the resulting topology of a simple clause $c_j = x_i$). We further create a list of nodes $v_1, v_2, \ldots v_l$ with edges (v_1, t), $(v_2, t) \ldots (v_l, t)$, and (v_1, v_2), $(v_2, v_3) \ldots (v_{l-1}, v_l)$. In addition, for each c_j node, create an edge (v_l, c_j). The final topology of a simple clause $c_j = x_i$ is shown in Figure 8b. The x_i nodes have no preference, each $\overline{x_i}$ node has preferences $\overline{x_i} x_i t \succ \overline{x_i} t \succ \overline{x_i} x_i s t$, each c_j node prefers any path of length 2 to those longer than 2. For any $1 \leq i \leq l$, v_i prefers the path $v_i v_{i+1} \ldots v_l p$ to the path $v_i t$, for any path p starting at a c_j node to t of length 3. It is straightforward to verify that the reduction is poly-time.

If C is satisfiable by an assignment A, consider the preference completion where a x_i node

prefers the direct path $x_i t$ to the path $x_i s t$ iff x_i is true in A. The preference of $\overline{x_i}$ ensures that among x_i and $\overline{x_i}$, only one can take a path of length 1 and the other will take a path of length 2. Consequently, each c_j node has a path of length 2 (through the literal that satisfies c_j in A), and hence each v_i node will take the path $v_i t$. The length of the path linearization is 2. On the other hand, if C is not satisfiable, for any path linearization, define an assignment where x_i is true iff the node x_i prefers the path $x_i t$ to $x_i s t$. At least one c_j node is not satisfied by this assignment, and hence every out-going neighbor of c_j has a path of length 2, and c_j must end up taking a path of length 3. Then, the node v_l can take the path through this c_j, and each v_i will take the path through v_{i-1}. Consequently, v_1 will end up with a path of length more than l. Since l can be made $\Omega(n)$ in this construction, distinguishing whether the minimum length of a path linearization is 2 or $\Omega(n)$ implies satisfiability of C. □

4.7.2. Approximate the Path-Length Minimization Problem

Theorem 4.7 establishes that PLM is hard to approximate to within a factor of $\Omega(n)$. However, note that this hardness result relies on the length of the longest path in the preference being large (that is, $\Omega(n)$). We complement the hardness result by the positive result below.

Theorem 4.8. *There is a poly-time algorithm that for any linearizable network $N\langle G(V,E), L, t\rangle$, outputs a path linearization with length at most $(OPT + l)$, where OPT is the minimum length of a path linearization of N, and l is the length of the longest path in L. Moreover, in the path linearization the algorithm outputs, every node only has preference over at most $(|L| + 1)$ paths.*

We say a path p is *compatible* with a state S, if for any edge (u, v) on p, the state of u in S, $S(u)$, is either $\perp$ (i.e., no edge is selected) or (u, v). A path p is said to be *fully compatible* with S if for any edge (u, v) on p, $S(u) = (u, v)$. A fully compatible path in S is always available. In addition, we said a node v has *stabilized* at a state S if v will not change its state under any schedule.

Proof. To find a path linearization with the properties stated in the theorem, we first create a spine (see Definition 4.11) for the given network $\langle G(V, E), L, t\rangle$, and argue that the spine guarantees a unique stable state S for a subset T of nodes where (i) for each node v in T, the path of v in the state S belongs to the spine, and (ii) for any node u in $V \setminus T$, no path specified in the preference of u, i.e., L_u, is compatible with S. Since all paths in the graph induced by S belong to the spine, the longest path among them has length at most l.

For the remaining nodes, since for any node v in $V \setminus T$, no path in L_v is compatible with S, any path p (starting at v) compatible with S can essentially be made the best path of v in S by letting p be less preferred than any path (starting at v) in the spine, but better than the rest. To find a path compatible with S for each node in $V \setminus T$ such that the length of the longest path is bounded, we treat all nodes in T as a 'super sink', and perform a BFS from the super sink to the remaining nodes. Consequently, every path in the resulting graph would be a combination of a path induced by S and a path in the BFS tree. Since the depth of the BFS tree is a lower bound of the minimum length of a path linearization by adding the path of each node v in the resulting graph to the preference of v, we achieve a stable state where path-length are at most $OPT + l$. For every node v, since every path in L can add at most one new path from v to t to the spine, v will have preferences on at most $(|L| + 1)$ paths. It remains to show how to find such a state S and a set of nodes T. We establish the following lemma.

Lemma 4.7.1. *Given a network with a spine, where the original preference is replaced by the spine, in any state S, if T is a subset of nodes that have stabilized, then either (i) there exists a node $v \in V \setminus T$, where the best compatible (with S) path in L_v (if any) is fully compatible, or (ii) for any $v \in V \setminus T$, no path in L_v is compatible with S.*

Proof. We prove by contradiction. For simplicity, a (fully) compatible path always refers to a path (fully) compatible to the state S. Suppose that for any node $v \in V \setminus T$, the best compatible path in L_v (if any) is not fully compatible, and there exists a node $v_0 \in V \setminus T$ who has compatible paths in L_{v_0}, and the best compatible path is p_0. By our assumption,

p_0 is not fully compatible. Hence, there exists a sub-path of p_0 denoted by v_1q_0 where q_0 is a path to t induced by S and $v_1 \notin T$ (i.e., the shortest sub-path to t not induced by S). Since the path v_1q_0 is on the spine, it is a path in the preference of v_1, L_{v_1} that is fully compatible. By our assumption, v_1q_0 is not the best compatible path in L_{v_1}. Consider the best compatible path p_1 of L_{v_1}, and using the same process, we can find a sub-path of p_1, vq_1, where the best compatible path of v_2 is p_2. Eventually, the process must reach the same node twice and the paths $\{p_i, v_{i+1}q_i\}$ forms a dispute wheel, which contradicts the fact that the network is linearizable. □

Then, starting from $T = \{t\}$ and an empty state S, we can repeatedly apply Lemma 4.7.1, find nodes whose best fully compatible path is available (which will be selected eventually), add them to T, and update S accordingly, until for any remaining node, no path in the preferences is compatible with S. □

4.8. Conclusions and Future Work

Partially specified preferences match the way that router configuration is envisaged: definite policy is established at coarse granularity, and further tie-breaking decisions are essentially arbitrary. In this chapter, we studied convergence time for partial routing preference systems and introduced the notion of linearizability to connect partial preference systems to earlier work on routing correctness and convergence.

We formally established that the routing convergence time is exacerbated by the complexity of the routing policy. This demonstrates that in order to improve the convergence speed of BGP, we must think not only about the router implementations, the network environment, and the protocol design, but also about the nature of the specified policies. In particular, our result in Section 4.5 shows that the real source of slow convergence is not the number of alternative paths, nor the length of paths, nor the presence of cycles, but the ability to express 'fine-grained' route preferences. Even if preferences are only allowed to be based on the two-hop neighborhood, exponential convergence time is possible.

On the positive side, restricting the preference system *does* help, though consequently, nodes might be only allowed to specify preference over either the next hop or a few short paths. Our work on 1-hop preference systems generalizes the work of Schapira et al. [100]; they additionally require that path preferences follow the guidelines of Gao and Rexford [52]. The next-hop restriction is in fact a reasonable one, since it matches a typical first-cut BGP policy—setting local preference based on the neighbor's identity.

In other related work, Fabrikant et al. [47] consider a braid-like network that may encounter exponentially many improvements before convergence, using essentially a $\langle 4, 3 \rangle$-preference system. Our work complements theirs by settling how much restriction one needs to place on the preference system in order to guarantee poly-time convergence.

We further consider the problem of finding linearization with desired properties and present our initial study for the property of minimizing the length of the longest path in the stable state. There are many other interesting and important properties to be considered such as minimizing congestion, minimizing delay etc, and it is an interesting direction of future work to study these properties for linearization. Another interesting direction of future work is to investigate even more general classes of partial preference systems and study their rapid convergence.

CHAPTER 5 : Matchings in the Simultaneous Communication Model

So far, we have introduced our work on the double oral auction (Chapter 2) and the interdomain routing problem (Chapter 4). In both problems, there are selfish players acting following their own myopic incentive, and we focused on understanding the question of how long it takes for the players to reach an agreement. The stochastic matching problem (Chapter 3) concerns the problem of finding a maximum matching when information about the input graph is distributed among multiple players.

In this chapter, we will still focus on the problem of finding a maximum matching, while in a different computational model, namely, the simultaneous communication model[1]. As we will elaborate later in this chapter, the simultaneous communication model has received significant attention recently, mainly due to its connection to the streaming model of computation. We first formally define the simultaneous communication model and the maximum matching problem in this model. Then, we present a randomized protocol and a deterministic protocol for this model, and establish their approximation ratio along with the total communication required. Finally, we generalize out deterministic protocol to a multi-round variant, which establishes a distinction between simultaneous communication and back and forth communication. We will start with formally defining the model and discussing our result.

5.1. Our Results and Related Work

We now formally defining the distributed model of computation considered in this chapter. In the *multi-party communication model*, the input (in our case, an input graph) is *adversarially* partitioned across k players and the goal is to design a *protocol* such that the players can jointly compute a function of the original input (in our case, an approximate matching). We distinguish between two possible ways of partitioning the input graph: in the *edge partition model*, each player holds a subset of edges of an input (multi-) graph

[1]The full paper of the work can be found in [16]. We would like to thank Michael Kapralov and David Woodruff for helpful discussions.

while in the *vertex partition model* the input graph must be bipartite and each player holds a distinct subset of vertices on the left together with all their adjacent edges. Two measures of complexity are considered, namely the *total communication*, which is simply the total size of all messages sent to the coordinator, and the *per-player communication* which is the maximum size of the messages sent by every player. We study both *deterministic* and *randomized* protocols (with *public coins*), and solely focus on *simultaneous protocols*, where every player simultaneously sends a message to a *coordinator* who outputs the final answer. Note that simultaneous protocols, in addition to their aforementioned connection to turnstiles algorithms, are indeed more preferable in distributed settings since they are naturally round-efficient [110].

Related Work

Matching in the multi-party communication model was previously studied under different variations [45, 10, 71, 69]. Huang *et al.* [71] focused on the k-party *message-passing* model (with two-way player-to-player communication) and gave a tight bound of $\Theta\left(\frac{nk}{\alpha^2}\right)$ on the total communication required to compute an α-approximate matching for both vertex partition and edge partition models. This result immediately implies a lower bound of $\Theta\left(\frac{nk}{\alpha^2}\right)$ for simultaneous protocols. However, the protocol in their upper bound is not simultaneous, and thus far it was not known if this lower bound is achievable by simultaneous protocols. The work of [45, 10] considers the bipartite matching problem in the n-party vertex partition model (i.e., every player holds a single vertex on the left). In particular, [45] showed that a protocol in which every player simultaneously sends a random incident edge achieves an $O(\sqrt{n})$-approximation, which matches the lower bound of [71]. The authors further studied deterministic protocols and showed an essentially tight bound on per-player communication of $\Theta(n^{1-\epsilon})$ bits for achieving an n^{ϵ}-approximation. However, these results do not directly generalize to an arbitrary number of players.

Our Results

For the vertex partition model, we show that the lower bound of $\Omega(nk/\alpha^2)$ proved for the message-passing model in [71] is achievable via (the weaker class of) simultaneous protocols, as long as $\alpha \geq \sqrt{k}$.

Theorem 5.1. *There exists a randomized simultaneous protocol that for any $k > 1$, computes an $O(\alpha)$-approximate matching in expectation in the k-party vertex partition model, while using total communication of (i) $O(nk/\alpha^2)$ when $\sqrt{k} \leq \alpha \leq k$, and (ii) $\widetilde{O}(n/\alpha)$ when $\alpha > k$.*

Since $\Omega(n/\alpha)$ is always a lower bound on the total communication, our protocol immediately achieves the optimal communication bound for any $\alpha > k$. Moreover, when $\sqrt{k} \leq \alpha \leq k$ and in the meaningful regime where $\alpha \leq \sqrt{n}$, we have $nk/\alpha^2 \geq k$, and hence the total communication is indeed $O(nk/\alpha^2)$, which matches the lower bound of [71].

The core idea in our simultaneous protocol is to send a "random matching" from each player to the coordinator. Note that for the case where $k = n$, since each player only has one vertex on the left, a random matching degrades to a random neighbor (as is used by [45]). However, for arbitrary k, simply sending random neighbors does not result in a protocol with good approximation guarantees. Indeed, we concentrate the bulk of our efforts on both finding a proper definition of random matchings for our purpose, and exploring their underlining structures. We show that picking a random order of the vertices on the right and computing a maximal matching following this order gives a suitable definition of a random matching. Our proof is based on a proper decomposition of the random orders, which allows us to define multiple *independent* events that were originally based on the same random order.

We should point out that by Theorem 5.1 and the aforementioned lower bound of [71], for $\alpha \geq \sqrt{k}$, allowing interaction between the coordinator and the players, compared to simultaneous protocols, does not lead to better performance (under the same communication bound). This is in contrast to the exponential communication gap between simultaneous

protocols and interactive protocols established in [45]. The main reason comes from the differences in the modeling assumptions: [45] focuses on the blackboard model where only the communication from the players to the coordinator (i.e., writing on the blackboard) is counted, and players can read the blackboard (which may contain $\Omega(n)$ bits) without additional communication cost. Consequently, one round of interaction in the blackboard model might lead to $\Omega(nk)$ communication in the k-party communication model.

Similar to [45], we also study the power of deterministic protocols and establish a tight bound of $\Theta(nk/\alpha)$ for the total communication for the case of $\alpha \geq \sqrt{k}$. This generalizes the bounds in [45] to arbitrary values of k.

Theorem 5.2. *For* deterministic *simultaneous protocols that compute an $O(\alpha)$-approximate matching in the k-party vertex partition model for any $\alpha^2 \geq k > 1$, the total communication of $O(nk/\alpha)$ is* sufficient.

Theorem 5.2 is achieved by a novel protocol whereby each player repeatedly finds maximum matchings that matches distinct sets of vertices on the right, and sends all matchings to the coordinator. We further extend our deterministic protocol to a multi-round variant which allows us to break the barrier of $\alpha \geq \sqrt{k}$.

Organization: The rest of this chapter is organized as follows. After introducing some preliminaries in Section 5.2, we present our randomized simultaneous protocols to prove Theorem 5.1 in Section 5.3. Then, we present our deterministic protocol in Section 5.4, and its multi-round variant in Section 5.5. Finally, we conclude in Section 5.6 with some further directions.

5.2. Preliminaries

Cauchy-Schwarz for multiple vectors.

Claim 5.2.1. *For any k vectors $x_1, \ldots, x_k \in \mathbb{R}^n$ it holds that:*

$$\left(\sum_{i=1}^{n}\prod_{j=1}^{k} x_{j,i}\right)^k \leq \prod_{j=1}^{k}\sum_{i=1}^{n} x_{j,i}^k$$

Proof. Proof is by induction on k. Note that for $k = 2$ the claim corresponds to the Cauchy-Schwarz inequality so the base holds. By Hölder's inequality we have:

$$\begin{aligned}
\left(\sum_{i=1}^{n}\prod_{j=1}^{k} x_{j,i}\right)^k &\leq \left(\left(\sum_{i=1}^{n}\left(\prod_{j=1}^{k-1} x_{j,i}\right)^{\frac{k}{k-1}}\right)^{\frac{k-1}{k}}\left(\sum_{i=1}^{n} x_{k,i}^k\right)^{\frac{1}{k}}\right)^k \\
&= \left(\sum_{i=1}^{n}\left(\prod_{j=1}^{k-1} x_{j,i}\right)^{\frac{k}{k-1}}\right)^{k-1}\left(\sum_{i=1}^{n} x_{k,i}^k\right) \\
&= \left(\sum_{i=1}^{n}\prod_{j=1}^{k-1} x_{j,i}^{\frac{k}{k-1}}\right)^{k-1}\left(\sum_{i=1}^{n} x_{k,i}^k\right) \\
&\leq \left(\prod_{j=1}^{k-1}\sum_{i=1}^{n} x_{j,i}^k\right)\left(\sum_{i=1}^{n} x_{k,i}^k\right) \\
&= \prod_{j=1}^{k}\sum_{i=1}^{n} x_{j,i}^k,
\end{aligned}$$

where the second inequality follows from the inductive hypothesis. □

Notation. Throughout this chapter, we will use opt to denote the size of a maximum matching in the input graph G.

5.3. Randomized Protocols

In this section, we establish Theorem 5.1 by presenting a simultaneous protocol for k players to compute an $O(\alpha)$-approximate matching for any $\alpha \geq \sqrt{k}$ using $O(nk/\alpha^2)$ total communication in the vertex partition model. We should note that technically, similar to the case for dynamic graph streams, since n/α could be the size of the target matching, the lower

bound of the total communication should be $\Omega(\max\{nk/\alpha^2, n/\alpha\})$. Consequently, when $\alpha \geq k$, $\Omega(n/\alpha)$ becomes the lower bound and when $\alpha \leq k$, $\Omega(nk/\alpha^2)$ is the lower bound. We give a protocol for each regime. The first regime is easier since at least one player contains $1/k$ fraction (which is at least $1/\alpha$ fraction) of any fixed optimum matching. The latter case is much more challenging and it is indeed the main contribution of this section.

5.3.1. A protocol with $\mathbf{O(nk/\alpha^2)}$ communication for $\sqrt{\mathbf{k}} \leq \alpha \leq \mathbf{k}$

We introduce the randomized protocol $\mathcal{P}^{rand}$. As we discussed in the previous section, the key of $\mathcal{P}^{rand}$ is to let every player to send a *random matching*, which we achieve in $\mathcal{P}^{rand}$ through randomly prioritizing the vertices in R when finding a matching.

Algorithm 4: $\mathcal{P}^{rand}$: a randomized $O(\alpha)$-approximation simultaneous protocol with $O(\frac{nk}{\alpha^2})$ communication (for $\sqrt{k} \leq \alpha \leq k$).

Input : A bipartite graph $G(L, R, E)$ in the simultaneous vertex-partition model with k players.

Output: A matching M of G with size $\Omega(\text{opt}/\alpha)$.

1. Let L_i be the vertices in L that belong to the i-th player P_i, and let $l_i = |L_i|$.
2. For each player P_i independently:
   (a) Pick a random permutation $\pi^{(i)}$ of the vertices in R.
   (b) Use $\pi^{(i)}$ to construct a matching M_i as follows: Enumerate the vertices v in R according to the order $\pi^{(i)}$, and match v with any unmatched neighbor if one exists.
   (c) Send the first $\left\lceil \frac{l_i \cdot k}{\alpha^2} \right\rceil$ edges of M_i to the coordinator.
3. The coordinator finds a maximum matching M among all received edges.

We first bound the total communication of $\mathcal{P}^{rand}$. Since each player only sends a matching of size at most $\left\lceil \frac{l_i \cdot k}{2\alpha^2} \right\rceil$, the total communication is $O(nk/\alpha^2 + k)$. Note that in the meaningful regime where $\alpha \leq \sqrt{n}$, $nk/\alpha^2 \geq k$, and the total communication is indeed $O(nk/\alpha^2)$. In the rest of this section, we show that the protocol $\mathcal{P}^{rand}$ outputs an α-approximate matching, and hence prove Theorem 5.1.

Fix a maximum matching M^* of G (of size opt). Let opt_i be the number of edges in M^* that belong to the i-th player P_i. A vertex $v \in R$ is said to be *good* for P_i if v is matched in M^* by an edge in P_i. The vertices in R that are *not good* for P_i are said to be *bad* for

P_i. A few remarks are in order.

Remark 5.3.1. *(a) Each player P_i will send at least $\left\lceil \frac{opt_i \cdot k}{2\alpha^2} \right\rceil$ ($\leq \left\lceil \frac{opt_i}{2} \right\rceil$ since $\alpha \geq \sqrt{k}$) edges. This is because $\mathcal{P}^{rand}$ can find a maximal matching in P_i, where the size of a maximum matching in P_i is at least opt_i. As it turns out, it suffices for us to only consider the first $\left\lceil \frac{opt_i \cdot k}{2\alpha^2} \right\rceil$ edges sent by P_i. (b) Without loss of generality, assume $opt_i < n/100$ for each player, since otherwise P_i will send a matching of size $\frac{opt_i \cdot k}{2\alpha^2} \geq \frac{opt}{200\alpha}$ (since $k \geq \alpha$), which is an $O(\alpha)$-approximation.*

One key component of our analysis is to decompose picking a random permutation $\pi^{(i)}$ into three *independent* components. $\pi^{(i)}_{pos}$: randomly pick opt_i positions in $[n]$ for placing the good vertices. We will refer to the picked positions as the *good positions* and the rest as *bad positions*; $\pi^{(i)}_b$: pick a random permutation of the bad vertices; and $\pi^{(i)}_g$: pick a random permutation of the good vertices. Then, placing the good/bad vertices in the good/bad positions following the orders $\pi^{(i)}_g/\pi^{(i)}_b$ gives the random permutation $\pi^{(i)}$. Observe that the three components $\pi^{(i)}_{pos}$, $\pi^{(i)}_b$, and $\pi^{(i)}_g$ are independent of each other, and hence events defined on different components are independent, which significantly simplifies the analysis. Moreover, we should note that, of course, each player does not know which vertices are good or bad, and hence the decomposition is only for the sake of analysis. We are now ready to prove that $\mathcal{P}^{rand}$ achieves an $O(\alpha)$-approximation.

Proof. (Proof of Theorem 5.1) Define $\mathcal{E}^*_i$ to be the event (on $\pi^{(i)}_b$) that, between L_i and the first n/α (bad) vertices in $\pi^{(i)}_b$, the maximum matching size is at least $\frac{\text{opt}_i \cdot k}{3\alpha^2}$. We partition the players into two types based probability of this event: Type 1 are players with $\Pr(\mathcal{E}^*_i) \geq 1/2$ and Type 2 are the rest. Let T_1 (resp. T_2) be the set of players that are Type 1 (resp. Type 2). Note that the type of each player only depends on the structure of his input graph and not the protocol.

In the following, we consider the case where players in T_1 contain at least opt/2 edges of M^* (Lemma 5.3.2) and the complement case where players in T_2 contain at least opt/2 edges of M^* (Lemma 5.3.4), separately.

Lemma 5.3.2. *If $\sum_{i\in T_1} opt_i \geq opt/2$, then $\mathrm{E}[|M|] = \Omega(opt/\alpha)$.*

Proof. Let $k_1 = |T_1|$. Without loss of generality, assume that the Type 1 players are $P_1, P_2, \ldots, P_{k_1}$ and the protocol $\mathcal{P}^{rand}$ is executed for these k_1 players following this specific order. Define S_i (for any $i \in [k_1]$) to be the set of the distinct vertices in R that are matched (and sent) by at least one of the first i players. To simplify the presentation, we further define $S_0 = \emptyset$. Then $\mathrm{E}[|M|] \geq \mathrm{E}[|S_{k_1}|]$ since each vertex in S_{k_1} is matched with a distinct vertex in L. We will prove that for any $i \in [k_1]$, if the size of S_{i-1} is at most $\frac{\mathrm{opt}}{30\alpha}$, then the i-th player will match a large number of new vertices in R in expectation. Formally,

Lemma 5.3.3. *For any integer $i \in [k_1]$ we have, $\mathrm{E}\left[|S_i \setminus S_{i-1}| \middle| |S_{i-1}| \leq \frac{opt}{30\alpha}\right] \geq 0.49 \cdot (\frac{opt_i \cdot k}{6\alpha^2} - \frac{opt}{15\alpha^2})$.*

Suppose we have Lemma 5.3.3 and define $\mathcal{E}'$ as the event that there exists an $i \in [k_1]$ where $|S_{i-1}| > \frac{\mathrm{opt}}{30\alpha}$. Note that if $\mathcal{E}'$ happens then $|S_{k_1}| > \frac{\mathrm{opt}}{30\alpha}$. Hence,

$$
\begin{aligned}
&\mathrm{E}[|S_{k_1}|] \\
=&\mathrm{E}\left[|S_{k_1}| \middle| \mathcal{E}'\right] \cdot \Pr\left(\mathcal{E}'\right) + \mathrm{E}\left[|S_{k_1}| \middle| \overline{\mathcal{E}'}\right] \cdot \Pr\left(\overline{\mathcal{E}'}\right) \\
\geq&\frac{\mathrm{opt}}{30\alpha} \cdot \Pr\left(\mathcal{E}'\right) + \sum_{i\in[k_1]} \mathrm{E}\left[|S_i \setminus S_{i-1}| \middle| |S_{i-1}| \leq \frac{\mathrm{opt}}{30\alpha}\right] \cdot \Pr\left(\overline{\mathcal{E}'}\right) \\
\geq&\frac{\mathrm{opt}}{30\alpha} \cdot \Pr\left(\mathcal{E}'\right) + 0.49 \sum_{i\in[k_1]} \left(\frac{\mathrm{opt}_i \cdot k}{6\alpha^2} - \frac{\mathrm{opt}}{15\alpha^2}\right) \cdot \Pr\left(\overline{\mathcal{E}'}\right) && \text{(by Lemma 5.3.3)} \\
\geq&\frac{\mathrm{opt}}{30\alpha} \cdot \Pr\left(\mathcal{E}'\right) + 0.49 \left(\frac{\mathrm{opt} \cdot k}{12\alpha^2} - \frac{\mathrm{opt} \cdot k}{15\alpha^2}\right) \cdot \Pr\left(\overline{\mathcal{E}'}\right) && \text{(since } \textstyle\sum_{i\in[k_1]} \mathrm{opt}_i \geq \mathrm{opt}/2\text{)} \\
=&\frac{\mathrm{opt}}{30\alpha} \cdot \Pr\left(\mathcal{E}'\right) + \Omega\left(\frac{\mathrm{opt}}{\alpha}\right) \Pr\left(\overline{\mathcal{E}'}\right) = \Omega\left(\frac{\mathrm{opt}}{\alpha}\right) && \text{(since } k \geq \alpha\text{)}
\end{aligned}
$$

Hence $\mathrm{E}[|M|] = \Omega(\mathrm{opt}/\alpha)$. We now prove Lemma 5.3.3.

Proof. (Proof of Lemma 5.3.3) We need to lower bound the expected number of new vertices in R that are matched by P_i compare to S_{i-1}. For any $\beta \geq 1$, define $g_\beta^{(i)}$ to be the random variable (on $\pi_{pos}^{(i)}$) counting the number of good positions that appear in the first $1/\beta$ fraction

of $[n]$. We will consider the joint event that $\mathcal{E}_i^*$ happens and the number of good positions that appear in the first $2/\alpha$ fraction is at most n/α (i.e., $g_{\alpha/2}^{(i)} \leq n/\alpha$).

$$\begin{aligned}
&\mathrm{E}\left[|S_i \setminus S_{i-1}| \middle| |S_{i-1}| \leq \frac{\mathrm{opt}}{30\alpha}\right] \\
\geq &\mathrm{E}\left[|S_i \setminus S_{i-1}| \middle| |S_{i-1}| \leq \frac{\mathrm{opt}}{30\alpha}, \mathcal{E}_i^*, g_{\alpha/2}^{(i)} \leq n/\alpha\right] \cdot \Pr\left(\mathcal{E}_i^*, g_{\alpha/2}^{(i)} \leq n/\alpha\right)
\end{aligned}$$

Since $\mathcal{E}_i^*$ is defined on $\pi_b^{(i)}$ and $g_{\alpha/2}^{(i)}$ is defined on $\pi_{pos}^{(i)}$, they are independent (due to the decomposition of $\pi^{(i)}$). We know that Type 1 players have $\Pr(\mathcal{E}_i^*) \geq 1/2$, so we only need to bound $\Pr\left(g_{\alpha/2}^{(i)} \leq n/\alpha\right)$. Since $\mathrm{opt}_i < n/100$ (by Remark 5.3.1(b)), $\mathrm{E}\left[g_{\alpha/2}^{(i)}\right] < (2/\alpha) \cdot (n/100) = n/50\alpha$. By Markov inequality, $\Pr\left(g_{\alpha/2}^{(i)} \geq n/\alpha\right) \leq 1/50$. Hence,

$$\begin{aligned}
&\Pr\left(\mathcal{E}_i^*, g_{\alpha/2}^{(i)} \leq n/\alpha\right) \\
=&\Pr(\mathcal{E}_i^*) \cdot \Pr\left(g_{\alpha/2}^{(i)} \leq n/\alpha\right) \geq 1/2 \cdot 49/50 = 0.49
\end{aligned}$$

It remains to lower bound

$$\mathrm{E}\left[|S_i \setminus S_{i-1}| \middle| |S_{i-1}| \leq \frac{\mathrm{opt}}{30\alpha}, \mathcal{E}_i^*, g_{\alpha/2}^{(i)} \leq n/\alpha\right]$$

We only consider the first $2n/\alpha$ vertices of $\pi^{(i)}$, denoted by $\pi^{(i)}[2n/\alpha]$, and analyze two quantities. x: the expected number of vertices in $\pi^{(i)}[2n/\alpha]$ that are matched, and y: the expected number of vertices in $\pi^{(i)}[2n/\alpha]$ that belong to S_{i-1}. We will show that $x \geq \frac{\mathrm{opt}_i \cdot k}{6\alpha^2}$ and $y \leq \frac{\mathrm{opt}}{15\alpha^2}$. Since $x - y$ is a lower bound of the expected number of new vertices in R that are matched in P_i, this will complete the proof.

For the quantity x, $g_{\alpha/2}^{(i)} \leq n/\alpha$ implies that there are at least n/α bad vertices in $\pi^{(i)}[2n/\alpha]$, and by $\mathcal{E}_i^*$, there is a matching of size at least $\frac{\mathrm{opt}_i \cdot k}{3\alpha^2}$ between L_i and the first n/α bad vertices (and hence between L_i and $\pi^{(i)}[2n/\alpha]$). Since P_i will find a maximal matching, at least $\frac{\mathrm{opt}_i \cdot k}{6\alpha^2}$ vertices in $\pi^{(i)}[2n/\alpha]$ will be matched.

For the quantity y, since $|S_{i-1}| \leq \frac{\text{opt}}{30\alpha}$, and each vertex in S_{i-1} belongs to $\pi^{(i)}[2n/\alpha]$ with probability $2/\alpha$, the expected number of vertices in S_{i-1} that belong to $\pi^{(i)}[2n/\alpha]$ is at most $\frac{\text{opt}}{15\alpha^2}$. Therefore,

$$\mathrm{E}\left[|S_i \setminus S_{i-1}| \,\middle|\, |S_{i-1}| \leq \frac{\text{opt}}{30\alpha}\right] \geq 0.49 \cdot \left(\frac{\text{opt}_i \cdot k}{6\alpha^2} - \frac{\text{opt}}{15\alpha^2}\right)$$

□

□

We now analyze the case where opt/2 edges of M^* belong to the players in T_2.

Lemma 5.3.4. *If $\sum_{i \in T_2} opt_i \geq opt/2$, then $\mathrm{E}[|M|] = \Omega(opt/\alpha)$.*

Proof. We will show for the Type 2 players that $\Omega(1/\alpha)$ fraction of the good vertices will be matched in expectation. Recall that for any $\beta \geq 1$, $g_\beta^{(i)}$ is the random variable (on $\pi_{pos}^{(i)}$) counting the number of good positions that appear in the first $1/\beta$ fraction of $[n]$. Then $\mathrm{E}\left[g_\beta^{(i)}\right] = \text{opt}_i/\beta$. Define $gm^{(i)}$ to be the random variable (on $\pi^{(i)}$) for the number of good vertices that are matched and sent to the coordinator by P_i. Since the size of M is at least the sum of $gm^{(i)}$, our goal is to lower bound $\mathrm{E}\left[gm^{(i)}\right]$. We establish the following key lemma.

Lemma 5.3.5. *For any player P_i in T_2, any integer $r \geq 1$, and any $\pi_{pos}^{(i)}$ with $g_\alpha^{(i)} = r$, $\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} = r\right] \geq \frac{1}{4}\min\left\{r, \left\lceil \frac{opt_i \cdot k}{6\alpha^2} \right\rceil\right\}$, where the expectation is taken over $\pi_b^{(i)}$ and $\pi_g^{(i)}$.*

Note that $\frac{1}{4}\min\left\{r, \left\lceil \frac{\text{opt}_i \cdot k}{6\alpha^2} \right\rceil\right\} \geq \frac{1}{4}$, and sometimes, we will directly use $\frac{1}{4}$ as a lower bound of the target expectation when applying Lemma 5.3.5. We first demonstrate how to use Lemma 5.3.5 to prove $\mathrm{E}[|M|] = \Omega(\text{opt}/\alpha)$. To see this, we need to further partition the Type 2 players into two sub-types, where for Type 2a: $\text{opt}_i/\alpha \geq 1$, and for Type 2$b$: $\text{opt}_i/\alpha < 1$. Let the set of players that are in Type 2a (resp. Type 2b) be T_{2a} (resp. T_{2b}). We consider these two sub-types separately.

Lemma 5.3.6. *If* $\sum_{i\in T_{2a}} opt_i \geq opt/4$, *then* $\mathrm{E}\left[|M|\right] = \Omega(opt/\alpha)$.

Proof. Fix any player P_i in T_{2a}. We can lower bound the expectation of $gm^{(i)}$ as follows.

$$\mathrm{E}\left[gm^{(i)}\right] \geq \mathrm{E}\left[gm^{(i)}\middle| g_\alpha^{(i)} \geq \frac{\mathrm{opt}_i}{2\alpha}\right] \cdot \Pr\left(g_\alpha^{(i)} \geq \frac{\mathrm{opt}_i}{2\alpha}\right)$$

Since a good position appearing in the first n/α fraction of $[n]$ is negatively correlated to other good position appearing in the first n/α fraction of $[n]$, by Chernoff bounds, $\Pr\left(g_\alpha^{(i)} < \frac{\mathrm{opt}_i}{2\alpha}\right) \leq \frac{1}{e^{1/12}}$. Denote by c the constant $\left(1 - \frac{1}{e^{1/12}}\right)$. Hence

$$\mathrm{E}\left[gm^{(i)}\right] \geq \mathrm{E}\left[gm^{(i)}\middle| g_\alpha^{(i)} \geq \frac{\mathrm{opt}_i}{2\alpha}\right] \cdot c$$

By Lemma 5.3.5,

$$\begin{aligned}
&\mathrm{E}\left[gm^{(i)}\middle| g_\alpha^{(i)} \frac{\mathrm{opt}_i}{2\alpha}\right] \cdot c \\
\geq &\frac{1}{4} \min\left\{\frac{\mathrm{opt}_i}{2\alpha}, \left\lceil \frac{\mathrm{opt}_i \cdot k}{6\alpha^2} \right\rceil\right\} \cdot c \\
\geq &\frac{c}{4}\frac{\mathrm{opt}_i}{6\alpha} = \Omega\left(\frac{\mathrm{opt}_i}{\alpha}\right)
\end{aligned}$$

Therefore, summing over all players in T_{2a},

$$\begin{aligned}
&\mathrm{E}\left[\sum_{i\in T_{2a}} gm^{(i)}\right] \\
= &\sum_{i\in T_{2a}} \mathrm{E}\left[gm^{(i)}\right] \\
= &\Omega\left(\sum_{i\in T_{2a}} \frac{\mathrm{opt}_i}{\alpha}\right) = \Omega\left(\frac{\mathrm{opt}}{\alpha}\right)
\end{aligned}$$

□

Lemma 5.3.7. *If* $\sum_{i\in T_{2b}} opt_i \geq opt/4$, *then* $\mathrm{E}\left[|M|\right] = \Omega(opt/\alpha)$.

Proof. For any player P_i in T_{2b},

$$\mathrm{E}\left[gm^{(i)}\right] \geq \mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} \geq 1\right] \cdot \Pr\left(g_\alpha^{(i)} \geq 1\right)$$

Since $\mathrm{opt}_i/\alpha < 1$, the probability that no good position appears in the first n/α is

$$\begin{aligned}
&\Pr\left(g_\alpha^{(i)} = 0\right) \\
=&\binom{n - n/\alpha}{\mathrm{opt}_i} \Big/ \binom{n}{\mathrm{opt}_i} \\
=&\frac{(n - n/\alpha)! \cdot (n - \mathrm{opt}_i)!}{n! \cdot (n - n/\alpha - \mathrm{opt}_i)!} \\
=&\prod_{j=0}^{n/\alpha - 1} \frac{n - \mathrm{opt}_i - j}{n - j} \\
\leq&\left(\frac{n - \mathrm{opt}_i}{n}\right)^{n/\alpha} \\
\leq&\exp(-\frac{\mathrm{opt}_i}{n} \cdot \frac{n}{\alpha}) \\
=&e^{-\mathrm{opt}_i/\alpha} \\
\leq&1 - \frac{\mathrm{opt}_i}{2\alpha}
\end{aligned}$$

where the last inequality is because $e^{-x} \leq 1 - x/2$ for any $x \in [0, 1]$. Therefore,

$$\begin{aligned}
&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} \geq 1\right] \cdot \Pr\left(g_\alpha^{(i)} \geq 1\right) \\
\geq&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} \geq 1\right] \cdot \frac{\mathrm{opt}_i}{2\alpha}
\end{aligned}$$

By Lemma 5.3.5,

$$\begin{aligned}
&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} \geq 1\right] \cdot \frac{\mathrm{opt}_i}{2\alpha} \\
\geq&\frac{1}{4} \cdot \frac{\mathrm{opt}_i}{2\alpha} \\
=&\Omega\left(\frac{\mathrm{opt}_i}{\alpha}\right)
\end{aligned}$$

Summing over all players in T_{2b},

$$\begin{aligned}
&\mathrm{E}\left[\sum_{i\in T_{2b}} gm^{(i)}\right]\\
=&\Omega\left(\sum_{i\in T_{2b}} \frac{\mathrm{opt}_i}{\alpha}\right)\\
=&\Omega\left(\frac{\mathrm{opt}}{\alpha}\right)
\end{aligned}$$

□

Proof. (Proof of Lemma 5.3.5) We need to lower bound the expected number of good vertices that are matched. It suffices for us to only consider this expectation when the event $\mathcal{E}_i^*$ does not happen, i.e., the first n/α bad vertices only have a matching of size less than $\frac{\mathrm{opt}_i \cdot k}{3\alpha^2}$ to L_i.

$$\begin{aligned}
&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} = r\right]\\
\geq&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} = r, \overline{\mathcal{E}_i^*}\right] \cdot \Pr\left(\overline{\mathcal{E}_i^*}\right)\\
\geq&\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} = r, \overline{\mathcal{E}_i^*}\right] \cdot \frac{1}{2}
\end{aligned}$$

In the following, we claim that when enumerating each of the first $\min\left\{\left\lceil\frac{\mathrm{opt}_i \cdot k}{6\alpha^2}\right\rceil, r\right\}$ good positions in the first n/α, (a) P_i still has the budget to send one more edge, and moreover, (b) with probability at least $1/2$, $\pi_g^{(i)}$ picks a good vertex that has an unmatched neighbor.

To see property (a), when enumerating any of these good positions, the number of vertices in R that are matched is strictly less than $\frac{\mathrm{opt}_i \cdot k}{3\alpha^2}$ (which is an upper bound of the number of matched bad vertices) plus $\left\lceil\frac{\mathrm{opt}_i \cdot k}{6\alpha^2}\right\rceil - 1$ (which is an upper bound of the number of good vertices that have appeared). Since $\left\lceil\frac{\mathrm{opt}_i \cdot k}{6\alpha^2}\right\rceil - 1 \leq \frac{\mathrm{opt}_i \cdot k}{6\alpha^2}$, the total number of vertices in R that are matched is strictly less than $\frac{\mathrm{opt}_i \cdot k}{2\alpha^2}$. Since P_i can send at least $\left\lceil\frac{\mathrm{opt}_i \cdot k}{2\alpha^2}\right\rceil$ edges, the number of matching edges is strictly less than the budget, and hence P_i can send at least one more edge.

To see property (b), since, again, at most $\text{opt}_i k/3\alpha^2$ bad vertices are matched, and $\text{opt}_i k/6\alpha^2$ good vertices have appeared, at least $\text{opt}_i/2$ good vertices that have not appeared have the property that the vertices they are matched with in M^* are still unmatched. Hence $\pi_g^{(i)}$ assign a good vertex that can be matched with probability at least $1/2$. Therefore,

$$\mathrm{E}\left[gm^{(i)} \middle| g_\alpha^{(i)} = r, \overline{\mathcal{E}_i^*}\right] \cdot \frac{1}{2} \geq \frac{1}{4} \cdot \min\left\{\left\lceil \frac{\text{opt}_i \cdot k}{6\alpha^2} \right\rceil, r\right\}$$

□

□

□

5.3.2. A protocol with $\widetilde{\mathbf{O}}(\mathbf{n}/\alpha)$ communication for $\alpha \geq \mathbf{k}$

We introduce the randomized protocol $\mathcal{P}_2^{rand}$. Intuitively, in the regime where $\alpha \geq k$, it is always the case that at least one player will contain a matching of size at least $\text{opt}/k (\geq \text{opt}/\alpha)$, and hence, one can always obtain an α-approximation by asking every player to send a (locally) maximum matching. However, in the case where many players have a matching of size opt/α (e.g., the case of a complete bipartite graph), the total communication could be as large as nk/α. To resolve this issue, we design a self-sampling scheme which ensures that when multiple players has a large matching, only a handful of them would actually end up sending a large matching.

Claim 5.3.8. *The protocol $\mathcal{P}_2^{rand}$ uses $\widetilde{O}(n/\alpha)$ communication.*

Proof. Since there are only $O(\log n)$ different guesses of opt, we only need to show that for each guess, $\tilde{\text{opt}}$, the total communication is $\widetilde{O}(n/\alpha)$. Fix an $\tilde{\text{opt}}$, since a player will send at most $\tilde{\text{opt}}/\alpha$ edges to the coordinator, we only need to show that only $\widetilde{O}(n/\tilde{\text{opt}})$ players will pass the coin toss and send a matching to the coordinator. The expected number of

Algorithm 5: $\mathcal{P}_2^{rand}$: A randomized $O(\alpha)$-approximation simultaneous protocol with $\widetilde{O}(\frac{n}{\alpha})$ communication (for $\alpha \geq k$).

Input : A bipartite graph $G(L, R, E)$ in the simultaneous vertex-partition model with k players.

Output: A matching M of G with size $\Omega(\text{opt}/\alpha)$.

1. For each player P_i independently,
   (a) Let l_i be the number of vertices in L that are in P_i.
   (b) Guess the size of a maximum matching in G (i.e., opt) from $\{n/\alpha, n/(2\alpha), n/(4\alpha), \ldots, 1\}$.
   (c) For each guessed value of opt, denoted by $\tilde{\text{opt}}$, toss a biased coin and with probability $\min\left\{2l_i \log n/\tilde{\text{opt}}, 1\right\}$, find a maximum matching M_i and send the first (at most) $\tilde{\text{opt}}/\alpha$ edges of M_i to the coordinator.
2. The coordinator finds a maximum matching among all received edges.

players that passed the coin toss is

$$\sum_{i\in[k]} 2l_i \log n/\tilde{\text{opt}} = 2n \log n/\tilde{\text{opt}} \geq 2 \log n$$

By Chernoff bounds, with high probability, at most $\widetilde{O}(n/\tilde{\text{opt}})$ players passes the coin toss, and hence the total communication is $\widetilde{O}(n/\alpha)$. □

Claim 5.3.9. *The protocol $\mathcal{P}_2^{rand}$ outputs a matching of size $\Omega(opt/\alpha)$.*

Proof. If $\text{opt} \leq n/\alpha$, since at least one player (say P_i) contains a matching of size at least opt/k (which is at least opt/α), it suffices to show that P_i will send a matching of size at least n/α^2 (which is $\geq \text{opt}/\alpha$). When P_i guesses $\tilde{\text{opt}} = n/\alpha$, the probability that P_i passes the coin toss is at least $2l_i \log n/\tilde{\text{opt}} \geq 1$, and P_i will send a matching of size $\tilde{\text{opt}}/\alpha = n/\alpha^2$.

If $\text{opt} > n/\alpha$, we argue that when every player guesses an $\tilde{\text{opt}} \in [\text{opt}/2, \text{opt}]$, a matching of size $\Omega(\tilde{\text{opt}}/\alpha)$ (which is also $\Omega(\text{opt}/\alpha)$) will be sent to the coordinator. Fix a matching M^* in G of size $\tilde{\text{opt}}$ and consider the vertices in M that belong to L, denoted by L_{M^*}. L_{M^*} is partitioned across k players. We refer to any player that contains more than $\tilde{\text{opt}}/(2\alpha)$ vertices in L_{M^*} as *good*. Since the size of a maximum matching in any good player P_i is at least the number of vertices in L_M that belong to P_i, any good player that passes the

coin toss will send a matching of size at least $\tilde{\text{opt}}/(2\alpha)$ to the coordinator, and $\mathcal{P}_2^{rand}$ would then output a matching of size $\Omega(\tilde{\text{opt}}/\alpha)$. We only need to show at least one good player will pass the coin toss.

Since the total number of vertices in L_{M^*} that belong to the players that are not good is at most $k \cdot \tilde{\text{opt}}/(2\alpha) \leq \tilde{\text{opt}}/2$, at least $\tilde{\text{opt}}/2$ vertices in L_{M^*} belongs to the good players. The expected number of good players that passed the coin toss is at least

$$\sum_{P_i \text{ is good}} 2l_i \log n/\tilde{\text{opt}} \geq (\tilde{\text{opt}}/2)2 \log n/\tilde{\text{opt}} = \log n$$

By Chernoff bounds, with high probability, at least one of the good players will pass the coin toss. □

5.4. Deterministic Protocols

When $\alpha \geq k$, a simple protocol where every player finds a maximum matching and send the first n/α edges will achieve the required approximation and communication. Hence, here we consider the $\alpha < k$ case and introduce the deterministic protocol $\mathcal{P}^{det}$. Intuitively, unlike randomized protocols, deterministic protocols cannot achieve coordination through objects like "random matchings", which is the reason why we need an extra α factor in the communication. The protocol $\mathcal{P}^{det}$ adapts the approach of repeatedly finding maximum matchings for each player until either no more vertex (in R) can be matched, or enough edges has been collected.

If we denote by L_i the set of vertices in L that belong to the player P_i and by l_i the size of L_i, P_i will send at most $\lceil k/\alpha \rceil \cdot l_i$ edges to the coordinator. Hence the total amount of communication is $(\sum_{i \in [k]} l_i)\lceil k/\alpha \rceil = O(nk/\alpha)$. We now show that the protocol $\mathcal{P}^{det}$ outputs an $O(\alpha)$-approximate matching and hence proving part (i) in Theorem 5.2.

Proof. (Proof of Theorem 5.2, part (i)) Let M be a maximum matching of the edges received by the coordinator and opt be the size of a maximum matching in G. Denote the sets of

Algorithm 6: $\mathcal{P}^{det}$: A deterministic $O(\alpha)$-approximation simultaneous protocol with $O(\frac{nk}{\alpha})$ communication.

Input : A bipartite graph $G(L, R, E)$ in the simultaneous vertex-partition model with k players.

Output: A matching M of G with size $\Omega(\text{opt}/\alpha)$.

1. For each player P_i independently:
   (a) Find a maximum matching M_1 and remove the vertices in R that are matched in M_1; find a maximum matching M_2 in the remaining graph and remove the vertices in R that are matched in M_2; repeat for $\lceil k/\alpha \rceil$ times. We refer to the set of the edges found in this process as a *matching-cover*.
   (b) Send the matching-cover to the coordinator.
2. The coordinator outputs a maximum matching of all received edges.

vertices matched in M by A (in L) and B (in R). No edge between $L \setminus A$ (denoted by $\overline{A}$) and $R \setminus B$ (denoted by $\overline{B}$) is received by the coordinator. We argue that size of M is at least $\text{opt}/(12\alpha)$. Suppose, by contradiction, that $|M| = |A| = |B| < \text{opt}/(12\alpha)$.

Let M^* be a maximum matching between $\overline{A}$ and $\overline{B}$ in G. Since G has a matching of size opt while A and B each only have at most $\text{opt}/(12\alpha)$ vertices, the size of M^* is at least $\text{opt} - |A| - |B| \geq \text{opt} - 2\frac{\text{opt}}{12\alpha} > 5\text{opt}/6 > 3\text{opt}/4$. Denote the sets of vertices matched in M^* by A^* (in L) and B^* (in R). The vertices in A^* are partitioned between the k players. Denote the vertices in A^* that belong to the player P_i by A_i^*, and denote by n_i the size of A_i^*. Hence $\sum_{i\in[k]} n_i = |A^*| \geq 3\text{opt}/4$. Denote the number of vertices in A that belong to P_i by a_i. To simplify the presentation, we further assume for each player P_i, if $a_i \neq 0$, n_i is an integer multiple of a_i[2]. Let B_i^* be the set of vertices matched to A_i^* in M^*. We first make the following observation.

Claim 5.4.1. *For any player P_i, the j-th maximum matching found by P_i, for any $1 \leq j \leq \lceil k/\alpha \rceil$, must match at least $(n_i - ja_i)$ vertices in $A^*(i)$.*

Proof. Fix a player P_i. Since no edge between A_i^* and B_i^* is sent to the coordinator, if a vertex $v \in B_i^*$ is matched by the matching-cover of P_i, v must be matched to a vertex in A. Since the number of vertices in A that belong to P_i is a_i, each maximum matching found by

[2]This can be achieved by removing at most a_i vertices from A_i^* for each player P_i. Since $\sum_{i\in[k]} a_i \leq |A|$, the size of the remaining matching in M^* is still at least $\text{opt} - 2|A| - |B| > 3\text{opt}/4$.

P_i can only match at most a_i of the vertices in B_i^*. Hence when P_i is finding a maximum matching for the j-th time, at least $(n_i - ja_i)$ vertices in B_i^* are unmatched. Consider the corresponding $(n_i - ja_i)$ vertices in A_i^* that are matched with these $(n_i - ja_i)$ vertices in B_i^* in the matching M^* (denoted by A''). Since A'' is not matched with any vertex in $\overline{B}$, P_i must match A'' with a set of $(n_i - ja_i)$ vertices in B. □ □

As an immediate consequence of Claim 5.4.1, we can establish the following connection between n_i and a_i for each player.

Lemma 5.4.2. *For each player P_i,* $\min\{n_i/a_i, \lceil k/\alpha \rceil\} \cdot (n_i - a_i)/2 \leq |B|$.

Proof. In the rest of the proof we assume that $n_i \geq 1$ since otherwise the inequality holds trivially. Using Claim 5.4.1, P_i matches at least $(n_i - ja_i)$ vertices in $A^*(i)$ in the j-th matching. Since once a vertex v in B is matched, v will be deleted from the graph, these $(n_i - ja_i)$ vertices in B must be distinct for each j.

Let $\beta = \min\{n_i/a_i, \lceil k/\alpha \rceil\}$. Then for the first β times of finding maximum matching,

$$\begin{aligned} |B| &\geq \sum_{j \in [\beta]} (n_i - ja_i) \\ &= \frac{((n_i - a_i) + (n_i - \beta a_i))\,\beta}{2} \\ &\geq \frac{(n_i - a_i)\beta}{2} \end{aligned}$$

where the last inequality is due to $\beta \leq n_i/a_i$. □ □

To use Lemma 5.4.2, we partition the players into two groups. The group $\mathcal{G}_1$ contains players P_i where $n_i/a_i \leq \lceil k/\alpha \rceil$, and the group $\mathcal{G}_2$ contains the players where $n_i/a_i > \lceil k/\alpha \rceil$. Since at least one of the two groups contains at least half of the edges in M^*, we consider $\mathcal{G}_1$ containing half of M^* and $\mathcal{G}_2$ containing half of M^* separately and show that for either case $|M| \geq \text{opt}/(12\alpha)$.

If $\mathcal{G}_1$ contains half of the edges in M^*, i.e., $\sum_{i\in\mathcal{G}_1} n_i \geq |M^*|/2 \geq 3\text{opt}/8 > \text{opt}/4$, using Lemma 5.4.2, we have

$$(n_i/a_i)(n_i - a_i)/2 \leq |B| \,(= |A|)$$

which implies $a_i \geq n_i^2/(2|A|+n_i)$. Note that $n_i \leq |A|$ since otherwise the player P_i contains a matching of size larger than $|A|$, and the edges sent by P_i alone must contain a matching of size larger than $|A|$, which contradicts the fact that the coordinator outputs a maximum matching of size $|A|$. Therefore, $a_i \geq n_i^2/(3|A|)$. Summing over all players in $\mathcal{G}_1$, we have

$$\begin{aligned}(|A| \geq) \sum_{i\in\mathcal{G}_1} a_i &\geq \frac{\sum n_i^2}{3|A|} = \frac{\sum n_i^2 |\mathcal{G}_1|}{3|A||\mathcal{G}_1|} \\ &\geq \frac{(\sum n_i)^2}{3k|A|} \geq \frac{\text{opt}^2}{48k|A|}\end{aligned}$$

where the second inequality is by the Cauchy-Schwarz. Hence, $|A| \geq \text{opt}/(\sqrt{48k}) \geq \text{opt}/(4\sqrt{3}\alpha) > \text{opt}/(12\alpha)$, a contradiction.

If $\mathcal{G}_2$ contains half of the edges in M^*, i.e., $\sum_{i\in\mathcal{G}_2} n_i \geq |M^*|/2 \geq 3\text{opt}/8$, Using Lemma 5.4.2, we have:

$$\lceil k/\alpha \rceil (n_i - a_i)/2 \leq |B| \,(= |A|)$$

which implies $a_i \geq n_i - 2\alpha|A|/k$. Summing over all players in $\mathcal{G}_2$, we have

$$(|A| \geq) \sum_{i\in\mathcal{G}_2} a_i \geq \sum_{i\in\mathcal{G}_2} (n_i - 2\alpha|A|/k) \geq 3\text{opt}/8 - 2\alpha|A|$$

which implies $|A|(1+2\alpha) \geq 3\text{opt}/8$, and hence $|A| \geq \text{opt}/(8\alpha)$, a contradiction. □

5.5. Multi-Round Protocols

Consider the following direct extension of the simultaneous protocol $\mathcal{P}^{det}$.

The matchings found by the coordinator form a matching of the entire graph, and the total size of the matchings is less that n/α, since otherwise the protocol has already found

Algorithm 7: $\mathcal{P}^r$: An r-round $O(\alpha)$-approximation deterministic protocol for $\alpha \geq k^{\frac{1}{1+r}}$.

Input : A bipartite graph $G(L, R, E)$ in the simultaneous vertex-partition model with k players.

Output: A matching M of G with size $\Omega(\text{opt}/\alpha)$.

1. For $l = 1$ to r rounds:
   (a) In the l-th round, the sites and the coordinator will invoke $\mathcal{P}$: each site finds a matching-cover with parameter α in the *remaining graph* and send it to the coordinator. The coordinator finds a maximum matching M_l among all received edges.
   (b) The coordinator will send the matching M_l to each site, and the site will remove all vertices matched in M_l from its input.
2. The coordinator outputs $M = M_1 \cup M_2 \cup \ldots \cup M_r$ as the final matching.

an α-approximate matching. Therefore, the coordinator will use $O(kn/\alpha)$ communication total.

Theorem 5.3. *For any bipartite graph $G(L, R, E)$, in the vertex partition model with k sites, for any $\alpha \geq k^{\frac{1}{r+1}}$ and any constant $r \geq 1$, the r-round protocol $\mathcal{P}^r$ outputs a $O(\alpha)$-approximation to the maximum matching with $O(kn/\alpha)$ communication.*

Proof. It is convenient to consider the rounds from the last to the first. To simplify the notation, we use the term *reversed-round* where the l-th reversed-round refers to the $(r - l + 1)$-th round.

Denote the matching found by the coordinator in the l-th reversed-round by M_l, and further denote the sets of vertices in M_l by A_l (in L) and B_l (in R). Let $a_l(i)$ be the number of vertices in A_l that belongs to the i-th site P_i. Denote the total size of the M_l matchings (for $l \in [r]$) by S, and suppose towards a contradiction that S is less than $\text{opt}/(2^{r+2}\alpha)$. Since there is a matching of size opt in G, there exists a matching of size at least opt/2 that does not intersect with any matching M_l. We denote this matching by M^* and the vertices in M^* by A^* and B^*. Denote by $A^*(i)$ that vertices in A^* that belongs to P_i, and let $n^*(i) = |A^*(i)|$.

At a high level, our approach is to argue that for each site P_i, for each reversed-round $l \in [r]$, there are many vertices in $A^*(i)$ that have many edges to B_l. Since in the $(l+1)$-th

reversed-round (i.e., in the previous round), the coordinator receives no edge between $A^*(i)$ and B_l, we can establish that P_i did not send any edge between $A^*(i)$ and B_l implies that P_i sent even more edges between $A^*(i)$ and B_{l+1} in the $(l+1)$ reversed-round. Eventually, this increment of the number of edges between $A^*(i)$ and B_{l+1} leads to a contraction in the r-th reversed-round (i.e., the first round) on the size of B_r.

To formalize the argument, we consider the following invariant for each site

$$c_l(i) = 2^{-\left(\frac{(l+1)l}{2}-1\right)} \cdot \frac{{n^*}^l}{\Pi_{l'<l} a_{l'}}$$

We first establish the following lower bound (using $c_l(i)$) on the number of vertices in $A^*(i)$ that P_i must match when finding a maximum matching each time in each round.

Lemma 5.5.1. *For any site P_i and any $l \in [r]$, if for any $l' < l$, $\lceil k/\alpha \rceil \geq c_{l'}(i)/a_{l'}(i)$, then at the l-th reversed-round, when P_i (using protocol $\mathcal{P}$) find maximum matchings for the j-th time, for any $j \leq c_l(i)/a_l(i)$, the number of vertices in $A^*(i)$ that appear in the j-th matching is at least*

$$\left(n^*(i) - \frac{j a_l(i) n^*(i)}{c_l(i)}\right) / 2^{l-1}.$$

Proof. We fix a site P_i in $\mathcal{G}_1$ and simplify the notation by using n^* to denote $n^*(i)$, a_l to denote $a_l(i)$, and c_l to denote $c_l(i)$. We prove this by induction on the (reversed) rounds.

Base. For $l = 1$, which is the last round, $c_1 = n^*$. Using Claim 5.4.1, when finding the j-th maximum matching, at least $(n^* - ja_1)$ vertices in $A^*(i)$ will be matched. This completes the proof of the induction base.

Inductive step. Assume the induction property holds for the l-th reversed-round, we now prove for the $(l+1)$-th reversed-round.

In the following, suppose for any $l' \leq l$, $\lceil k/\alpha \rceil \geq c_{l'}/a_{l'}$. We first show that there exist at least $n^*/2^l$ vertices in $A^*(i)$ where each of them has at least $c_l/2a_l$ edges to B_l, and

then show how to use the existence of these high degree (w.r.t. B_l) vertices to prove the induction property.

Proposition 5.5.2. *There exists at least $n^*/2^l$ vertices in $A^*(i)$ where each of them has at least $c_l/2a_l$ edges to B_l.*

Proof. We first prove the existence of $n^*/2^l$ high degree (w.r.t. B_l) vertices. In the l-th reversed-round, P_i find maximum matching for c_l/a_l times, and the j-th time matches at least $(n^* - ja_l n^*/c_l)/2^{l-1}$ vertices from $A^*(i)$. Since P_i only matches vertices in $A^*(i)$ to vertices in B_l (hence forming degree w.r.t. B_l), we only need to show that the matchings P_i found in the l-th reversed-round match a set of $n^*/2^l$ vertices in $A^*(i)$ for at least $c_l/2a_l$ times. To see this, we show that the set of vertices in $A^*(i)$ that are matched in the j-th matching must also be matched in the $(j-1)$-th matching, and hence the first $c_l/2a_l$ matchings all hit a set of $(n^* - (c_l/2a_l)(a_l n^*/c_l))/2^{l-1} = n^*/2^l$ vertices. Suppose a vertex u is matched in the j-th time but not in the $(j-1)$-th time, then the match of u in the j-th time can be added to the maximum matching found in the $(j-1)$-th time and forms a larger matching, a contradiction. Therefore, there exists at least $n^*/2^l$ vertices in $A^*(i)$ where each of them has at least $c_l/2a_l$ edges to B_l. □

In the $(l+1)$-th reversed-round, consider the first c_l/a_l times of finding maximum matchings. The total number of vertices in R (in particular, the total number of vertices in B^* and B_l) that need to be covered is at least

$$n^* + \sum_{j \in [c_l/a_l]} \left(n^* - \frac{j a_l n^*}{c_l} \right) / 2^{l-1} \geq n^* + \frac{1}{2^{l-1}} \left(n^* - \frac{a_l n^*}{c_l} \right) \cdot \frac{c_l}{2a_l} = n^* + \frac{c_l n^*}{2^l a_l} - \frac{n^*}{2^l} \geq c_{l+1}$$

where the last inequality uses

$$c_l = \frac{2^{l+1} a_l}{n^*} \cdot c_{l+1}$$

We now ready to show that the j-th time matches at least $(n^* - ja_{l+1} n^*/c_{l+1})/2^l$ vertices in $A^*(i)$, hence proving the induction step.

When finding the maximum matching for the j-th time, at most ja_{l+1} vertices in B_l can be matched. For any vertex u that has degree at least $c_l/2a_l$ to B_l, we say the neighborhood of u in B_l is covered if all neighbors of u in B_l are matched in some of the first j matchings. As long the neighborhood of u in B_l is not covered, the maximum matching P_i found in the j-th time must also match u. Among the $n^*/2^l$ vertices in $A^*(i)$ that has degree at least $c_l/2a_l$ to B_l, the number of vertices whose neighborhood in B_l are covered is at most

$$\frac{ja_{l+1}}{c_l/2a_l} = \frac{2ja_{l+1}a_l}{c_l} = 2ja_{l+1}a_l \cdot \frac{n^*}{2^{l+1}a_l c_{l+1}} = \frac{ja_{l+1}n^*}{2^l c_{l+1}}$$

Hence, at least $(n^* - ja_{l+1}n^*/c_{l+1})/2^l$ many vertices in $A^*(i)$ will appear in the j-th matching. □

To use Lemma 5.5.1, similar to the single round case, we partition the sites into two groups. The group $\mathcal{G}_1$ contains the sites P_i where for every round $l \in [r]$, $\lceil k/\alpha \rceil \geq c_l(i)/a_l(i)$, and group $\mathcal{G}_2$ contains the sites where for at least one round $l \in [r]$, $\lceil k/\alpha \rceil < c_l(i)/a_l(i)$. One of the two groups contains more than half of the edges in M^*. In the following, we consider $\mathcal{G}_1$ containing at least half of M^* and $\mathcal{G}_2$ containing at least half of M^*, respectively.

Lemma 5.5.3. *If $\mathcal{G}_1$ contains more than half of M^*, then $S \geq opt/(2^r\alpha)$.*

Proof. Using Lemma 5.5.1, for the r-th reversed round (i.e., the first round), when P_i find maximum matchings for the j-th time, at least $(n^* - ja_r n^*/c_r)/2^{r-1}$ vertices in $A^*(i)$ must be matched with some distinct vertices in B_r. Since $\lceil k/\alpha \rceil \geq c_r/a_r$, for the first c_r/a_r times of finding maximum matchings,

$$|B_r| \geq \sum_{j \in [c_r/a_r]} \left(n^* - \frac{ja_r n^*}{c_r}\right)/2^{r-1} \geq \frac{1}{2^{r-1}}\left(n^* - \frac{a_r n^*}{c_r}\right)\frac{c_r}{2a_r} = \frac{c_r n^*}{2^r a_r} - \frac{n^*}{2^r} \geq c_{r+1} - \frac{n^*}{2^r}$$

Summing over all sites in $\mathcal{G}_1$ (to simplify the notation, all summations below are over sites in P_i),

$$(kS \geq) \, |\mathcal{G}_1| \, |B_r| \geq \sum \left(c_{r+1}(i) - \frac{n^*(i)}{2^r}\right)$$

Since $n^* < S$,

$$(2kS \geq) kS + \frac{kn^*}{2^r} \geq \frac{1}{2^{\frac{(r+1)r}{2}}} \sum \frac{n^*(i)^{r+1}}{\Pi_{l\in[r]} a_l(i)}$$

Using Cauchy-Schwarz inequality (the multiple vectors version, see Claim 5.2.1),

$$\sum \frac{n^*(i)^{r+1}}{\Pi_{l\in[r]} a_l(i)} = \left(\sum \frac{n^*(i)^{r+1}}{\Pi_{l\in[r]} a_l(i)}\right) \Pi_{l\in[r]} \left(\sum a_l(i)\right) / \left(\Pi_{l\in[r]} \sum a_l(i)\right)$$

$$\geq \left(\sum \left(\frac{n^*(i)^{r+1}}{\Pi_{l\in[r]} a_l(i)} \Pi_{l\in[r]} a_l(i)\right)^{\frac{1}{r+1}}\right)^{r+1} / \Pi_{l\in[r]} |A_l| \geq \left(\sum n^*(i)\right)^{r+1} / S^r \geq \left(\frac{\text{opt}}{4}\right)^{r+1} / S^r$$

where the last inequality is because $\mathcal{G}_1$ contains half of M^* where the size of M^* is at least opt/2. Hence

$$2kS \geq \frac{\text{opt}^{r+1}}{2^{\frac{(r+1)r}{2}} S^r}$$

which implies $S \geq \text{opt}/(2^r k^{\frac{1}{r+1}}) \geq \text{opt}/(2^r \alpha)$. □

Lemma 5.5.4. *If $\mathcal{G}_2$ contains more than half of M^*, then $S \geq opt/(2^{r+2}\alpha)$.*

Proof. For any $l \in [r]$, let $\mathcal{G}_2^l$ be the set of sites P_i in $\mathcal{G}_2$ where l is the smallest reversed-round (i.e., the latest round) that $\lceil k/\alpha \rceil < c_l(i)/a_l(i)$. Since for any $l' < \hat{l}$, $\lceil k/\alpha \rceil \geq c_{l'}(i)/a_{l'}(i)$ for all sites P_i in $\mathcal{G}_2^{\hat{l}}$, we can apply Lemma 5.5.1 to lower bound the size of $B_{\hat{l}}$. Formally, for any $l \in [r]$, for each P_i in $\mathcal{G}_2^{\hat{l}}$, and consider the $\lceil k/\alpha \rceil$ maximum matchings found by P_i,

$$(S \geq)|B_l| \geq \sum_{j\in[\lceil k/\alpha \rceil]} \left(n^*(i) - \frac{j a_l(i) n^*(i)}{c_l(i)}\right) / 2^{l-1} \geq \left(n^*(i) - \frac{a_l(i) n^*(i)}{c_l(i)}\right) \lceil k/\alpha \rceil / 2^r$$

Since $\lceil k/\alpha \rceil < c_l(i)/a_l(i)$ and $n^* < S$,

$$(2S \geq) S + \frac{a_l(i) n^*(i)}{c_l(i)} \lceil k/\alpha \rceil / 2^r \geq n^*(i)(k/\alpha)/2^r$$

Summing over all $l \in [l]$ and all sites in $\mathcal{G}_2^l$,

$$(2kS \geq) 2S \sum_{l \in [r]} \left| \mathcal{G}_2^l \right| \geq \left(\sum_{l \in [r]} \sum_{P_i \in \mathcal{G}_2^l} n^*(i) \right) (k/\alpha)/2^r \geq (\text{opt}/2)(k/\alpha)/2^r$$

Therefore,

$$S \geq \frac{\text{opt}}{2^{r+2}\alpha}$$

□

Hence, either $\mathcal{G}_1$ contains half of M^* or $\mathcal{G}_2$ contains half of M^*, $S \geq \text{opt}/(2^{r+2}\alpha)$. □

5.6. Conclusions and Future Work

In this chapter, we resolve the per-player simultaneous communication complexity for approximating matchings in the vertex partition model in the regime where $\alpha \geq \sqrt{k}$. We extend our protocol to multi-round which breaks the barrier of $\alpha \geq \sqrt{k}$. An interesting direction for future research is to obtain better understandings of matchings using multi-round protocols.

CHAPTER 6 : Data Provisioning

We have observed the power of data summarization for finding matchings in various settings (Chapter 3 and Chapter 5). In this chapter, we will see more usage of data summarization when handling distributed information. In particular, we will present our work in designing efficient schemes for data provisioning[1]. We first formally introduce the provisioning model and discuss the objective of our algorithms. Then, we present our provisioning schemes for numerical queries, logical queries, and the more general complex queries, which are practical queries combing logical, grouping, and numerical queries. We start with defining and motivating the query provisioning model.

6.1. Background

Query provisioning comes from the more general field of "what if analysis", which is a common technique for investigating the impact of decisions on outcomes in science or business. The main difference is that in our notation, data are distributed in multiple locations while for "what if analysis" data are typically assumed to be centralized. However, since the goal is to compute sketches of small size (rather than minimizing communication between players), one can always centralized the data first and then compute the sketch. In fact, all our algorithms for numerical queries (which is the main focus of this work) can be (essentially) directly executed in a distributed manner where the total communication is the same as the final sketch size. Therefore, for simplicity, in the rest of this chapter, we will directly focus on provisioning for "what if analysis".

"What if analysis" almost always involves a data analytics computation. Nowadays such a computation typically processes very large amounts of data and thus may be expensive to perform, especially repeatedly. An analyst is interested in exploring the computational impact of multiple *scenarios* that assume modifications of the input to the analysis problem. The general aim is to avoid repeating expensive computations for each scenario. For a given

[1]The full paper of this work can be found in [14]

problem, and starting from a given set of potential scenarios, we wish to perform just *one* possibly expensive computation producing a small *sketch* (i.e., a compressed representation of the input) such that the answer for any of the given scenarios can be derived rapidly from the sketch, without accessing the original (typically very large) input. We say that the sketch is "provisioned" to deal with the problem under any of the scenarios and following [41], we call the whole approach *provisioning*. Again, the goal of provisioning is to allow an analyst to efficiently explore a multitude of scenarios, using only the sketch and thus avoiding expensive recomputations for each scenario.

We apply the provisioning approach to queries that perform *in-database analytics* [70][2]. These are queries that combine *logical* components (relational algebra and Datalog), grouping, and *numerical* components (e.g., aggregates and quantiles). Other analytics are discussed under further work.

Abstracting away any data integration/federation, we will assume that the inputs are relational instances and that the scenarios are defined by a set of *hypotheticals*. We further assume that each hypothetical indicates the fact that certain tuples of an input instance are *retained* (other semantics for hypotheticals are discussed under further work).

A scenario consists of turning on/off each of the hypotheticals. Applying a scenario to an input instance therefore means keeping only the tuples retained by at least one of the hypotheticals that are turned on. Thus, a trivial sketch can be obtained by applying each scenario to the input, solving the problem for each such modified input and collecting the answers into the sketch. However, with k hypotheticals, there are *exponentially* (in k) many scenarios. Hence, even with a moderate number of hypotheticals, the size of the sketch could be enormous. Therefore, as part of the statement of our problem we will aim to provision a query by an algorithm that maps each (large) input instance to a *compact* (essentially size poly(k)) sketch.

[2]In practice, the MADlib project [2] has been one of the pioneers for in-database analytics, primarily in collaboration with Greenplum DB [1]. By now, major RDBMS products such as IBM DB2, MS SQL Server, and Oracle DB already offer the ability to combine extensive analytics with SQL queries.

Example. Suppose a large retailer has many and diverse sales venues (e.g., its own stores, its own web site, through multiple other stores, and through multiple other web retailers). An analyst working for the retailer is interested in learning, for each product in, say, "Electronics", a regression model for the way in which the *revenue* from the product depends on both a sales venue's *reputation* (assume a numerical score) and a sales venue *commission* (in %; 0% if own store). Moreover, the analyst wants to ignore products with small sales volume unless they have a large MSRP (manufacturer's suggested retail price). Usually there is a large (possibly distributed/federated) database that captures enough information to allow the computation of such an analytic query. For simplicity we assume in this example that the revenue for each product ID and each sales venue is in one table and thus we have the following query with a self-explanatory schema:

```
SELECT x.ProdID, LIN_REG(x.Revenue, z.Reputation, z.Commission) AS (B, A1, A2)
FROM  RevenueByProductAndVenue x
INNER JOIN Products y ON x.ProdID=y.ProdID
INNER JOIN SalesVenues z ON x.VenueID=z.VenueID
WHERE y.ProdCategory="Electronics" AND (x.Volume>100 OR y.MSRP>1000)
GROUP BY x.ProdID
```

The syntax for treating linear regression as a multiple-column-aggregate is simplified for illustration purposes in this example. Here the values under the attributes `B,A1,A2` denote, for each `ProdID`, the coefficients of the linear regression model that is learned, i.e., `Revenue = B + A1*Reputation + A2*Commission`.

A desirable what-if analysis for this query may involve hypotheticals such as retaining certain venue types, retaining certain venues with specific sales tax properties, retaining certain product types (within the specified category, e.g., tablets), and many others. Each of these hypotheticals can in fact be implemented as selections on one or more of the tables in the query (assuming that the schema includes the appropriate information). However, combining hypotheticals into scenarios is problematic. The hypotheticals overlap and thus

cannot be separated. With 10 (say) hypotheticals there will be $2^{10} = 1024$ (in practice at least hundreds) of regression models of interest for each product. Performing a lengthy computation for each one of these models is in total very onerous. Instead, we can *provision* the what-if analysis of this query since the query in this example falls within the class covered by our positive results.

6.2. Our Results and Related Work

Our goal is to characterize the feasibility of provisioning with sketches of *compact* size (see Section 6.3 for a formal definition) for a practical class of *complex queries* that consist of a *logical* component (relational algebra or Datalog), followed by a *grouping* component, and then by a *numerical* component (aggregate/analytic) that is applied to each group (a more detailed definition is given in Section 6.5).

The main challenge that we address, and the part where our main contribution lies, is the design of compact provisioning schemes for numerical queries, specifically linear (ℓ_2) regression and quantiles. Together with the usual count, sum and average, these are defined in Section 6.4 as queries that take a set of numbers or of tuples as input and return a number or a tuple of constant width as output. It turns out that if we expect exact answers, then none of these queries can be compactly provisioned. However, we show that compact provisioning schemes indeed exist for all of them if we relax the objective to computing near-exact answers (see Section 6.3 for a formal definition). The following theorem summarizes our results for numerical queries (see Section 6.4):

Theorem 6.1 (Informal)**.** *The* quantiles, count, and sum/average (of positive numbers) queries *can be compactly provisioned to provide (multiplicative) approximate answers to an arbitrary precision.*

It was shown in [14] that exact provisioning of these queries requires the sketch size to be exponential in the number of hypotheticals. This highlights the additional power of being able to produce approximate answers.

Our results on provisioning numerical queries can then be used for complex queries, as the following theorem summarizes (see Section 6.5):

Theorem 6.2 (Informal)**.** *Any complex query whose logical component is a* positive relational algebra *query can be compactly provisioned to provide an approximate answer to an arbitrary precision as long as its numerical component can be compactly provisioned for the same precision, and as long as the number of groups is not too large.*

It was also shown in [14] that having *negation* or *recursion* in the logical component requires the sketch size to be exponential in the number of hypotheticals. Therefore, further generalizing Theorem 6.2 would require completely new direction and perhaps new techniques.

Our techniques. At a high-level, our approach for compact provisioning can be described as follows. We start by building a sub-sketch for each hypothetical by focusing solely on the retained tuples of each hypothetical individually. We then examine these sub-sketches against each other and collect additional information from the original input to summarize the effect of appearance of other hypotheticals to each already computed sub-sketch. The first step usually involves using well-known (and properly adjusted) sampling or sketching techniques, while the second step, which is where we concentrate the bulk of our efforts, is responsible for gathering the information required for combining the sketches and specifically dealing with overlapping hypotheticals. Given a scenario, we answer the query by fetching the corresponding sub-sketches and merging them together; the result is a new sketch that act as sketch for the input consist of the *union* of the hypotheticals.

Related work. Our techniques for compact provisioning share some similarities with those used in data streaming and in the distributed computation models of [37, 36, 109] (see Section 6.6 for further discussion and formal separations), and in particular with *linear sketching*, which corresponds to applying a linear transformation to the input data to obtain the sketch. However, due to overlap in the input, our sketches are required to to be composable with the *union* operation (instead of the *addition* operation obtained by linear

sketches) and hence linear sketching techniques are not directly applicable.

Dealing with duplicates in the input (similar to the overlapping hypotheticals) has also been considered in streaming and distributed computation models (see, e.g., [35, 32]), which consider sketches that are "duplicate-resilient". Indeed, for simple queries like count, a direct application of these sketches is sufficient for compact provisioning (see Section 6.4.1). We also remark that the Count-Min sketch [34] can be applied to approximate quantiles even in the presence of duplication (see [32]), i.e., is duplicate-resilient. However, the approximation guarantee achieved by the Count-Min sketch for quantiles is only *additive* (i.e., $\pm\epsilon n$), in contrast to the stronger notion of *multiplicative* approximation (i.e., $(1 \pm \epsilon)$) that we achieve in this paper. To the best of our knowledge, there is no similar result concerning duplicate-resilient sketches for multiplicative approximation of quantiles, and existing techniques do not seem to be applicable for our purpose. Indeed one of the primary technical contributions of this paper is designing provisioning schemes that can effectively deal with overlapping hypotheticals for quantiles.

Further related work. *Provisioning*, in the sense used in this paper, originated in [41] together with a proposal for how to perform it taking advantage of provenance tracking. Answering queries under hypothetical updates is studied in [53, 18] but the focus there is on using a specialized warehouse to avoid transactional costs. (See also [41] for more related work.)

Estimating the number of distinct elements (corresponding to the count query) has been studied extensively in data streams [49, 9, 21, 73] and in certain distributed computation models [37, 36, 109]. For estimating quantiles, [85, 54, 58, 34, 59, 111] achieve an additive error of ϵn for an input of length n, and [66, 33] achieve a (stronger guarantee of) $(1 \pm \epsilon)$-approximation.

Organization: The rest of this chapter is organized as follows. We start by introducing the provisioning model formally and defining the necessary notation in Section 6.3. In

Section 6.4, we provide our results for numerical queries and prove Theorem 6.1. Section 6.5 contains our results for the complex queries and the proof of Theorem 6.2. We provide a comparison between the techniques for query provisioning and the distributed computation model in Section 6.6. Finally, we conclude with several future directions in Section 6.7.

6.3. Preliminaries

Hypotheticals. Fix a relational schema Σ. Our goal is to provision queries on Σ-instances. A *hypothetical* w.r.t. Σ is a computable function h that maps every Σ-instance I to a sub-instance $h(I) \subseteq I$. Formalisms for specifying hypotheticals are of course of interest (e.g., apply a selection predicate to each table in I) but we do not discuss them here because the results in this paper do not depend on them.

Scenarios. We will consider analyses (scenario explorations) that start from a finite set H of hypotheticals. A *scenario* is a non-empty set of hypotheticals $S \subseteq H$. The result of applying a scenario $S = \{h_1, \ldots, h_s\}$ to an instance I is defined as a sub-instance $I|_S = h_1(I) \cup \cdots \cup h_s(I)$. In other words, under S, if any $h \in S$ is said to be turned on (similarly, any $h \in H \setminus S$ is turned off), each turned on hypothetical h will retain the tuples $h(I)$ from I.

Definition 6.1 (Provisioning)**.** *Given a query Q, to* provision *Q means to design a pair of algorithms: (i) a **compression** algorithm that takes as input an instance I and a set H of hypotheticals, and outputs a data structure Γ called a **sketch**, and (ii) an **extraction** algorithm that for any scenario $S \subseteq H$, outputs $Q(I|_S)$ using only Γ (without access to I).*

To be more specific, we assume the compression algorithm takes as input an instance I and k hypotheticals $h_1, \ldots, h_k$ along with the sub-instances $h_1(I), \ldots, h_k(I)$ that they define. A hypothetical will be referred to by an index from $\{1, \ldots, k\}$, and the extraction algorithm will be given scenarios in the form of sets of such indices. Hence, we will refer to a scenario $S \subseteq H$ where $S = \{h_{i_1}, \ldots, h_{i_s}\}$ by abusing the notation as $S = \{i_1, \ldots, i_s\}$. Throughout the paper, we denote by k the number of hypotheticals (i.e. $k := |H|$), and by n the size of

the input instance (i.e., $n := |I|$).

We call such a pair of compression and extraction algorithms a *provisioning scheme*. The compression algorithm runs only once; the extraction algorithm runs repeatedly for all the scenarios that an analyst wishes to explore. We refer to the time that the compression algorithm requires as the *compression time*, and the time that extraction algorithm requires for each scenario as the *extraction time*.

The definition above is not useful by itself for positive results because it allows for trivial space-inefficient solutions. For example, the definition is satisfied when the sketch Γ is defined to be a copy of I itself or, as mentioned earlier, a scenario-indexed collection of all the answers. Obtaining the answer for each scenario is immediate for either case, but such a sketch can be prohibitively large as the number of tuples in I could be enormous, and the number of scenarios is exponential in $k = |H|$.

This discussion leads us to consider complexity bounds on the size of the sketches.

Definition 6.2 (Compact provisioning)**.** *A query Q can be* compactly *provisioned if there exists a provisioning scheme for Q that given any input instance I and any set of hypotheticals H, constructs a sketch of size poly$(k, \log n)$ bits, where $k := |H|$ and $n := |I|$.*

Even though the definition of compact provisioning does not impose any restriction on either the compression time or the extraction time, all our positive results in this paper are supported by (efficient) polynomial time algorithms. Note that this is *data-scenario complexity*: we assume the size of the query (and the schema) to be a constant but we consider dependence on the size of the instance and the number of hypotheticals.

Exact vs. approximate provisioning. Definition 6.2 focused on exact answers for the queries. While this is appropriate for, e.g., relational algebra queries, as we shall see, for queries that compute numerical answers such as aggregates and analytics, having the flexibility of answering queries approximately is essential for any interesting positive result.

Definition 6.3 (ϵ-provisioning)**.** *For any $0 < \epsilon < 1$, a query Q can be* ϵ-provisioned *if there*

exists a provisioning scheme for Q, whereby for each scenario S, the extraction algorithm outputs a $(1 \pm \epsilon)$ approximation of $Q(I|_S)$, where I is the input instance.

We say a query Q can be compactly *ϵ-provisioned if Q can be ϵ-provisioned by a provisioning scheme that, given any input instance I and any set of hypotheticals H, creates a sketch of size poly$(k, \log n, 1/\epsilon)$ bits.*

We emphasize that throughout this paper, we only consider the approximation guarantees which are *relative* (multiplicative) as opposed to the weaker notion of additive approximations. The precise definition of relative approximation guarantee will be provided for each query individually. Moreover, as expected, randomization will be put to good use in ϵ-provisioning. We therefore extend the definition to cover the provisioning schemes that use both randomization and approximation.

Definition 6.4. *For any $\epsilon, \delta > 0$, an* (ϵ, δ)-provisioning scheme *for a query Q is a provisioning scheme where both the compression and extraction algorithms are allowed to be randomized and the output* for every *scenario S is a $(1 \pm \epsilon)$-approximation of $Q(I|_S)$ with probability $1 - \delta$.*

An (ϵ, δ)-provisioning scheme is called compact *if it constructs sketches Γ of size only poly$(k, \log n, 1/\epsilon, \log(1/\delta))$ bits, has compression time that is poly$(k, n, 1/\epsilon, \log(1/\delta))$, and has extraction time that is poly$(|\Gamma|)$.*

In many applications, the size of the database is a very large number, and hence the exponent in the poly(n)-dependence of the compression time might become an issue. If the dependence of the compression time on the input size is essentially linear, i.e., $O(n) \cdot \text{poly}(k, \log n, 1/\epsilon, \log(1/\delta))$ we say that the scheme is *linear*. We emphasize that in all our positive results for queries with numerical answers we give compact (ϵ, δ)-linear provisioning schemes, thus ensuring efficiency in both running time and sketch size.

Complex queries. Our main target consists of practical queries that combine logical, grouping, and numerical components. In Section 6.5, we focus on *complex queries* defined

by a logical (relational algebra or Datalog) query that returns a set of tuples, followed by a group-by operation (on set of grouping attributes) and further followed by numerical query that is applied to each sets of tuples resulting from the grouping. This class of queries already covers many practical examples. We observe that the output of such a complex query is a set of p tuples where p is *the number of distinct values taken by the grouping attributes*. Therefore, the size of any provisioning sketches must also depend on p. We show (in Theorem 6.6) that a sketch for a query that involves grouping can be obtained as a collection of p sketches. Hence, if each of the p sketches is of compact size (as in Definitions 6.2 and 6.4) and the value p itself is bounded by $\text{poly}(k, \log n)$, then the overall sketch for the complex query is also of compact size. Note that p is typically small for the kind of grouping used in practical analysis queries (e.g., number of products, number of departments, number of locations, etc.). Intuitively, an analyst would have trouble making sense of an output with a large number of tuples.

Notation. For any integer $m > 0$, $[m]$ denotes the set $\{1, 2, \ldots, m\}$. The $\tilde{O}(\cdot)$ notation suppresses $\log\log(n)$, $\log\log(1/\delta)$, $\log(1/\epsilon)$, and $\log(k)$ factors. All logarithms are in base two unless stated otherwise.

6.4. Numerical Queries

In this section, we study provisioning of numerical queries, i.e., queries that output some (rational) number(s) given a set of tuples. In particular, we investigate aggregation queries including count, sum, average, and quantiles (therefore min, max, median, rank, and percentile), and as a first step towards provisioning database-supported machine learning, linear (ℓ_2) regression. We assume that the relevant attribute values are rational numbers of the form a/b where both a, b are integers in range $[-W, W]$ for some $W > 0$.

6.4.1. The Count, Sum, and Average Queries

In this section, we study provisioning of the count, sum, and average queries, formally defined as follows. The answer to the count query is the number of tuples in the input instance. For

the other two queries, we assume a relational schema with a binary relation containing two attributes: an *identifier (key)* and a *weight*. We say that a tuple x is smaller than the tuple y, if the weight of x is smaller than the weight of y. Given an instance I, the answer to the sum query (resp. the average query) is the *total weights* of the tuples (resp. the *average weight* of the tuples) in I.

We conclude this section by explaining the ϵ-provisioning schemes for the count, sum, and average queries. Formally,

Theorem 6.3 (ϵ-provisioning count)**.** *For any $\epsilon, \delta > 0$, there exists a compact (ϵ, δ)-linear provisioning scheme for the count query that creates a sketch of size $\tilde{O}(\epsilon^{-2}k(k + \log(1/\delta)))$ bits.*

Theorem 6.4 (ϵ-provisioning sum & average)**.** *For instances with positive weights, for any $\epsilon, \delta > 0$, there exists compact (ϵ, δ)-linear provisioning schemes for the sum and average queries, with a sketch of size $\tilde{O}(\epsilon^{-2}k^2 \log(n) \log(1/\delta) + k \log \log W)$ bits.*

We remark that the results in Theorems 6.3 and 6.4 are mostly direct application of known techniques and are presented here only for completeness.

The count query can be provisioned by using *linear sketches* for estimating the ℓ_0-norm (see, e.g., [73]) as follows. Consider each hypothetical $h_i(I)$ as an n-dimensional boolean vector $\mathbf{x}_i$, where the j-th entry is 1 iff the j-th tuple in I belongs to $h_i(I)$. For each $\mathbf{x}_i$, create a linear sketch (using $\widetilde{O}(\epsilon^{-2} \log n)$ bits of space) that estimates the ℓ_0-norm [73]. Given any scenario S, combine (i.e., add together) the linear sketches of the hypotheticals in S and use the combined sketch to estimate the ℓ_0-norm (which is equal to the answer of count).

Remark 6.4.1. *Note that we can directly use linear sketching for provisioning the count query since counting the duplicates once (as done by union) or multiple times (as done by addition) does not change the answer. However, this is* not *the case for other queries of interest like quantiles and regression and hence linear sketching is not directly applicable for them.*

Here, we also describe a self-contained approach for ϵ-provisioning the count query with a slightly better dependence on the parameter n ($\log\log n$ instead of $\log n$).

We use the following fact developed by [21] in the streaming model of computation to design our scheme. For a bit string $s \in \{0,1\}^+$, denote by trail(s) the number of trailing 0's in s. Given a list of integers $A = (a_1, \ldots, a_n)$ from the universe $[m]$, a function $g : [m] \to [m]$, and an integer $t > 0$, the $\langle t, g\rangle - trail$ of A is defined as the list of the t smallest trail($g(a_i)$) values (use binary expression of $g(a_i)$), where the duplicate elements in A are counted only once.

Lemma 6.4.2 ([21]). *Given a list $A = (a_1, \ldots, a_n)$, $a_i \in [m]$ with F_0 distinct elements, pick a* random pairwise independent hash function $g : [m] \to [m]$, *and let $t = \lceil 256\epsilon^{-2} \rceil$. If r is the largest value in the $\langle t, g\rangle - trail$ of A and $F_0 \geq t$, then with probability at least $1/2$, $t \cdot 2^r$ is a $(1 \pm \epsilon)$ approximation of F_0.*

We now define our (ϵ, δ)-linear provisioning scheme for the count query.

Algorithm 8: Compression algorithm for the count query.

Given an input instance I, a set H of hypotheticals, and an $\epsilon > 0$:
1. Assign each tuple in I with a unique number (an identifier) from the set $[n]$.
2. Let $t = \lceil 256\epsilon^{-2} \rceil$. Pick $\lceil k + \log(1/\delta) \rceil$ random pairwise independent hash functions $\{g_j : [n] \to [n]\}_{i=1}^{\lceil k+\log(1/\delta)\rceil}$. For each hash function g_j, create a sub-sketch as follows.
   (a) Compute the $\langle t, g_j\rangle - trail$ over the identifiers of the tuples in each $h_i(I)$.
   (b) Assign a new identifier to any tuple that accounts for at least one value in the $\langle t, g_j\rangle - trail$ of any h_i, called the *concise identifier*.
   (c) For each hypothetical h_i, **record** each value in the $\langle t, g_j\rangle - trail$ along with the concise identifier of the tuple that accounts for it.

Algorithm 9: Extraction algorithm for the count query.

Given a scenario S, for each hash function g_j, we use the concise identifiers to compute the union of the $\langle t, g_j\rangle - trail$ of the hypotheticals that are turned on by S. Let r be the t-th smallest value in this union, and compute $t \cdot 2^r$. Output the median of these $t \cdot 2^r$ values among all the hash functions.

We call a sketch created by the above compression algorithm a CNT-Sketch. For each hash function g_j and each $h_i(I)$, we record concise identifiers and the number of trailing 0's ($O(\log\log n)$ bits each) of at most t tuples. Since at most $t \cdot k$ tuples will be assigned with a

concise identifier, $O(\log(t \cdot k))$ bits suffice for describing each concise identifier. Hence the total size of a CNT-Sketch is $\tilde{O}(\epsilon^{-2}k \cdot (k + \log(1/\delta)))$ bits. We now prove the correctness of this scheme.

Proof of Theorem 6.3. Fix a scenario S; for any picked hash function g_i, since the t-th smallest value of the union of the recorded $\langle t, g_i \rangle - trail$, r, is equal to the t-th smallest value in the $\langle t, g_i \rangle - trail$ of $I|_S$. Hence, by Lemma 6.4.2, with probability at least $1/2$, $t \cdot 2^r$ is a $(1 \pm \epsilon)$ approximation of $|I|_S|$. By taking the median over $\lceil k + \log(1/\delta) \rceil$ hash functions, the probability of failure is at most $\delta/2^k$. If we take union bound over all 2^k scenarios, with probability at least $1 - \delta$, all scenarios could be answered with a $(1 \pm \epsilon)$ approximation. □

We now state the scheme for provisioning the sum query; the schemes for the sum and the count queries together can directly provision the average query, which finalizes the proof of Theorem 6.4. We use and extend our CNT-Sketch to ϵ-provision the sum query.

Algorithm 10: Compression algorithm for the sum query.

Given an instance I, a set H of hypotheticals, and two parameters $\epsilon, \delta > 0$, let $\epsilon' = \epsilon/4$, $t = \lceil \log_{1+\epsilon'}(n/\epsilon') \rceil$, and $\delta' = \frac{\delta}{k(t+1)}$.

1. Let $p = \left\lceil \log_{(1+\epsilon')} W \right\rceil$ and for any $l \in [p]$, let $\overline{w}_l = (1+\epsilon')^l$. For each $l \in [p]$, define a set of k new hypotheticals $H_l = \{h_{l,1}, h_{l,2}, \ldots, h_{l,k}\}$, where $h_{l,i}(I) \subseteq h_i(I)$ and contains the tuples whose weights are in the interval $[\overline{w}_l, \overline{w}_{l+1})$.
2. For each hypothetical h_i, let w be the largest weight of the tuples in $h_i(I)$, and find the index γ such that $\overline{w}_\gamma \leq w < \overline{w}_{\gamma+1}$. **Record** γ, and discard all the tuples in $h_i(I)$ with weight less than $\overline{w}_{\gamma-t}$.[3] Consequently, all the remaining tuples of $h_i(I)$ lie in the $(t+1)$ intervals $\{[\overline{w}_l, \overline{w}_{l+1})\}_{l=\gamma-t}^{\gamma}$. We refer to this step as the *pruning step*.
3. For each l, denote by $\hat{H}_l$ the resulting set of hypotheticals after discarding the above small weight tuples from H_l (some hypotheticals might become empty). For each of the $\hat{H}_l$ that contains at least one non-empty hypothetical, run the compression algorithm that creates a CNT-Sketch for I and $\hat{H}_l$, with parameters ϵ' and δ'. **Record** each created CNT-Sketch.

We call a sketch created by the above provisioning scheme a SUM-Sketch. Since for every hypothetical we only record the $(t+1)$ non-empty intervals, by an amortized analysis, the size of the sketch is $\tilde{O}(\epsilon^{-2}k^2 \log(n) \log(1/\delta) + k \log\log W)$ bits. We now prove the correctness

Algorithm 11: Extraction algorithm for the sum query.

Given a scenario S, for any interval $[\overline{w}_l, \overline{w}_{l+1})$ with a recorded CNT-Sketch, compute the estimate of the number of tuples in the interval, denoted by $\tilde{n}_l$. Output the summation of the values $(\overline{w}_{l+1} \cdot \tilde{n}_l)$, for l ranges over all the intervals $[\overline{w}_l, \overline{w}_{l+1})$ with a recorded CNT-Sketch.

of this scheme.

Proof of Theorem 6.4. For now assume that we do not perform the pruning step (line (2) of the compression phase). For each interval $[\overline{w}_l, \overline{w}_{l+1})$ among the $\lceil \log_{1+\epsilon'} W \rceil$ intervals, the CNT-Sketch outputs a $(1 \pm \epsilon')$ approximation of the total number of tuples whose weight lies in the interval. Each tuple in this interval will be counted as if it has weight $\overline{w}_{l+1}$, which is a $(1+\epsilon')$ approximation of the original tuple weight. Therefore, we can output a $(1 \pm \epsilon')^2$ approximation of the correct sum.

Now consider the original SUM-Sketch with the pruning step. We need to show that the total weight of the discarded tuples is negligible. For each hypothetical h_i, we discard the tuples whose weights are less than $\overline{w}_{\gamma-t}$, while the largest weight in $h_i(I)$ is at least $\overline{w}_\gamma$. Therefore, the total weight of the discarded tuples is at most

$$n\overline{w}_{\gamma-t} \leq \frac{nw_\gamma}{(1+\epsilon')^{\lceil \log_{(1+\epsilon')}(n/\epsilon') \rceil}} \leq \epsilon' \overline{w}_\gamma$$

Since whenever h_i is turned on by a given scenario, the sum of the weights is at least $\overline{w}_\gamma$, we lose at most ϵ' fraction of the total weight by discarding those tuples from $h_i(I)$. To see why we only lose an ϵ' fraction over all the hypotheticals (instead of $\epsilon' k$), note that at most n tuples will be discarded in the whole scenario, hence the n in the inequality $n\overline{w}_{\gamma-t} \leq \epsilon' \overline{w}_\gamma$ can be amortized over all the hypotheticals. □

6.4.2. The Quantiles Query

In this section, we study provisioning of the quantiles query. We again assume a relational schema with just one binary relation containing attributes identifier and weight. For any

instance I and any tuple $x \in I$, we define the *rank* of x to be the number of tuples in I that are smaller than or equal to x (in terms of the weights). The output of a quantiles query with a given parameter $\phi \in (0, 1]$ on an instance I is the tuple with rank $\lceil \phi \cdot |I| \rceil$. Finally, we say a tuple x is a $(1 \pm \epsilon)$-approximation of a quantiles query whose correct answer is y, iff the rank of x is a $(1 \pm \epsilon)$-approximation of the rank of y.

We now turn to prove the main theorem of this section, which argue the existence of a compact scheme for ϵ-provisioning the quantiles. We emphasize that the approximation guarantee in the following theorem is *multiplicative*.

Theorem 6.5 (quantiles)**.** *For any $\epsilon, \delta > 0$, there exists a compact (ϵ, δ)-linear provisioning scheme for the quantiles query that creates a sketch of size $\tilde{O}(k\epsilon^{-3} \log n \cdot (\log(n/\delta) + k)(\log W + k))$ bits.*

We should note that in this theorem the parameter ϕ is only provided in the extraction phase[4]. Our starting point is the following simple lemma first introduced by [66].

Lemma 6.4.3 ([66])**.** *For any list of* unique *numbers $A = (a_1, \ldots, a_n)$ and parameters $\epsilon, \delta > 0$, let $t = \lceil 12\epsilon^{-2} \log(1/\delta) \rceil$; for any* target rank *$r > t$, if we independently sample each element with probability t/r, then with probability at least $1 - \delta$, the rank of the t-th smallest sampled element is a $(1 \pm \epsilon)$-approximation of r.*

The proof of Lemma 6.4.3 is an standard application of the Chernoff bound and the main challenge for provisioning the quantiles query comes from the fact that hypotheticals overlap. We propose the following scheme which addresses this challenge.

We call a sketch created by the above compression algorithm a QTL-Sketch. It is straightforward to verify that the size of QTL-Sketch is as stated in Theorem 6.5. We now prove the correctness of the above scheme.

Proof of Theorem 6.5. Given an instance I and a set H of hypotheticals , we prove that

[4]We emphasize that we gave a lower bound for the easier case in terms of provisioning (ϕ given at compression phase and disjoint hypotheticals), and an upper bound for the harder case (ϕ given at extraction phase and overlapping hypotheticals).

Algorithm 12: Compression algorithm for the quantiles query.

Given an instance I, a set H of hypotheticals, and two parameters $\epsilon, \delta > 0$, let $\epsilon' = \epsilon/5$, $\delta' = \delta/3$, and $t = \lceil 12\epsilon'^{-2}(\log(1/\delta') + 2k + \log(n)) \rceil$.

1. Create and **record** a CNT-Sketch for I and H with parameters ϵ' and δ'.
2. Let $\left\{ r_j = (1+\epsilon')^j \;\middle|\; j \in \left[0..\left\lceil \log_{(1+\epsilon')} n \right\rceil\right] \right\}$. For each r_j, create the following sub-sketch individually.
3. If $r_j \leq t$, for each hypothetical h_i, **record** the r_j smallest chosen tuples in $h_i(I)$. If $r_j > t$, for each hypothetical h_i, choose each tuple in $h_i(I)$ with probability t/r_j, and **record** the $\lceil (1+3\epsilon') \cdot t \rceil$ smallest tuples in a list $T_{i,j}$. For each tuple x in the resulting list $T_{i,j}$, **record** its *characteristics vector* for the set of the hypotheticals, which is a k-dimensional binary vector $(v_1, v_2, \ldots, v_k)$, with value 1 on v_l whenever $x \in h_l(I)$ and 0 elsewhere.

Algorithm 13: Extraction algorithm for the quantiles query.

Suppose we are given a scenario S and a parameter $\phi \in (0, 1]$. In the following, the rank of a tuple always refers to its rank in the sub-instance $I|_S$.

1. Denote by $\tilde{n}$ the output of the CNT-Sketch on S. Let $\tilde{r} = \phi \cdot \tilde{n}$, and find the index γ, such that $r_\gamma \leq \tilde{r} < r_{\gamma+1}$.
2. If $r_\gamma \leq t$, among all the hypotheticals turned on by S, take the union of the recorded tuples and output the r_γ-th smallest tuple in the union.
3. If $r_\gamma > t$, from each h_i turned on by S, and each tuple x recorded in $T_{i,\gamma}$ with a characteristic vector $(v_1, v_2, \ldots, v_k)$, *collect* x iff for any $l < i$, either $v_l = 0$ or $h_l \notin S$. In other words, a tuple x recorded by h_i is taken only when among the hypotheticals that are turned on by S, i is the smallest index s.t. $x \in h_i(I)$. We will refer to this procedure as the *deduplication*. Output the t-th smallest tuple among all the tuples that are collected.

with probability at least $1-\delta$, for every scenario S and every parameter $\phi \in (0,1]$, the output for the quantiles query on $I|_S$ is a $(1\pm\epsilon)$-approximation. Fix a scenario S and a parameter $\phi \in (0,1]$; the goal of the extraction algorithm is to return a tuple with rank in range $(1\pm\epsilon)$ of the *queried rank* $\lceil \phi \cdot |I|_S| \rceil$. Recall that in the extraction algorithm, $\tilde{n}$ is the output of the CNT-Sketch. Therefore, $\tilde{r} = \phi\tilde{n}$ is a $(1\pm\epsilon')$ approximation of the queried rank, and r_γ with $r_\gamma < \hat{r} \leq (1+\epsilon')r_\gamma$ is a $(1\pm 3\epsilon')$ approximation of the rank. In the following, we argue that the tuple returned by the extraction algorithm has a rank in range $(1\pm\epsilon')\cdot r_\gamma$, and consequently is a $(1\pm\epsilon)$-approximation answer to the quantiles query.

If $r_\gamma \leq t$, the r_γ-th smallest tuple of $I|_S$ is the r_γ-th smallest tuple of the union of the recorded tuples $T_{i,\gamma}$ for $i \in S$. Therefore, we obtain a $(1\pm\epsilon)$ approximation in this case.

If $r_\gamma > t$, for any $i \in S$, define $\hat{T}_i$ to be the list of all the tuples sampled from $h_i(I)$ in the compression algorithm (instead of maintaining the $(1+3\epsilon')t$ tuples with smallest ranks). Hence $T_{i,\gamma} \subseteq \hat{T}_i$. If we perform the deduplication procedure on the union of the tuples in $\hat{T}_i$ for $i \in S$ and denote the resulting list T^*, then every tuple in $I|_S$ has probability exactly t/r_γ to be taken into T^* (for any tuple, only the appearance in the smallest index $h_i(I)$ could be taken). Hence, by Lemma 6.4.3, with probability at least $1-\delta'/2^{k+\log(n)}$, the rank of the t-th tuple of T^* is a $(1\pm\epsilon')$-approximation of the rank r_γ. In the following, we assume this holds.

The extraction algorithm does not have access to $\hat{T}_i$'s. Instead, it only has access to the list $T_{i,\gamma}$, which only contains the $(1+3\epsilon')t$ tuples of $\hat{T}_i$ with the smallest ranks. We show that with high probability, the union of $T_{i,\gamma}$ for $i \in S$, contains the first t smallest tuples from T^*. Note that for any $i \in S$, if the largest tuple x in $\hat{T}_i \cap T^*$ is in $T_{i,\gamma}$, then all tuples in $\hat{T}_i \cap T^*$ are also in $T_{i,\gamma}$ (e.g. the truncation happens after the tuple x). Hence, we only need to bound the probability that x is truncated, which is equivalent to the probability of the following event: more than $(1+3\epsilon')t$ tuples in $h_i(I)$ are sampled in the compression phase for the rank r_γ (with probability t/r_γ).

Let L_i be the set of all the tuples in $h_i(I)$ which are smaller than x. Rank of x is less than or equal to the rank of the t-th smallest tuple in T^*, which is upper bounded by $(1+\epsilon')r_\gamma$. Hence, $|L_i| \leq (1+\epsilon)r_\gamma$. For any $j \in h_i(I)$, define a binary random variable y_j, which is equal to 1 iff the tuple with rank j is sampled and 0 otherwise. The expected number of the tuples that are sampled from L is then $E[\sum_{j\in L} y_j] = |L| \cdot (t/r_\gamma) \leq (1+\epsilon')t$.

Using Chernoff bound, the probability that more than $(1+3\epsilon')t$ tuples from L are sampled is at most $\frac{\delta'}{2^{2k+\log(n)}}$. If we take union bound over all the hypotheticals in S, with probability at least $1 - \frac{\delta'}{2^{k+\log(n)}}$, for all h_i, the largest tuple in $\hat{T}_i \cap T^*$ is in $T_{i,\gamma}$, which ensures that the rank of the returned tuple is a $(1 \pm \epsilon)$-approximation of the queried rank.

Finally, since there are only n different values for $\phi \in (0, 1]$ which results in different answers, applying union bound over all these n different values of ϕ and 2^k possible scenarios, with probability at least $(1-2\delta')$, the output of the extraction algorithm is a $(1+\epsilon)$ approximation of the quantiles query. Since the failure probability of creating the CNT-Sketch is at most δ', with probability $(1-3\delta') = (1-\delta)$ the QTL-Sketch successes. □

Extensions. By simple extensions of our scheme, many variations of the quantiles query can be answered, including outputting the rank of a tuple x, the percentiles (the rank of x divided by the size of the input), or the tuple whose rank is Δ larger than x, where $\Delta > 0$ is a given parameter. As an example, for finding the rank of a tuple x, we can find the tuples with ranks approximately $\{(1+\epsilon)^l\}$, $l \in [\lceil \log_{(1+\epsilon)} n \rceil]$, using the QTL-Sketch, and among the found tuples, output the rank of the tuple whose weight is the closest to the weight of x.

6.5. Complex Queries

We study the provisioning of queries that combine logical components (relational algebra and Datalog), with grouping and with the numerical queries that we studied in Section 6.4.

We start by defining a class of such queries and their semantics formally. For the purposes

of this paper, a *complex query* is a triple $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ where Q_L is a relational algebra or Datalog query that outputs some relation with attributes $\overline{AB}$ for some $\overline{B}$, $G_{\overline{A}}$ is a *group-by* operation applied on the attributes $\overline{A}$, and Q_N is a numerical query that takes as input a relation with attributes $\overline{B}$. For any input I let $P = \Pi_{\overline{A}}(Q_L(I))$ be the $\overline{A}$-relation consisting of all the distinct values of the grouping attributes. We call the size of P the *number of groups* of the complex query. For each tuple $\overline{u} \in P$, we define $\Gamma_{\overline{u}} = \{\overline{v} \mid \overline{uv} \in Q_L(I)\}$. Then, the output of the complex query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ is a set of tuples $\{\langle \overline{u}, Q_N(\Gamma_{\overline{u}}) \rangle \mid \overline{u} \in P\}$.

6.5.1. Positive, Non-Recursive Complex Queries

In the following, we give compact provisioning results for the case where the logical component is a *positive relational algebra* (i.e., SPJU) query. It will be convenient to assume a different, but equivalent, formalism for these logical queries, namely that of *unions of conjunctive queries* (UCQs)[5]. We review quickly the definition of UCQs. A *conjunctive query* (CQ) over a relational schema Σ is of the form $ans(\overline{x}) \ :- \ R_1(\overline{x}_1), \ldots, R_b(\overline{x}_b)$, where atoms $R_1, \ldots, R_b \in \Sigma$, and the *size* of a CQ is defined to be the number of atoms in its body (i.e., b). A union of conjunctive query (UCQ) is a finite union of some CQs whose heads have the same schema.

In the following theorem, we show that for any complex query, where the logical component is a positive relational algebra query, compact provisioning of the numerical component implies compact provisioning of the complex query itself, provided the number of groups is not too large.

Theorem 6.6. *For any complex query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ where Q_L is a* UCQ*, if the numerical component Q_N can be compactly provisioned (resp. compactly ϵ-provisioned), and if the number of groups is bounded by poly$(k, \log n)$, then the query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ can also be compactly provisioned (resp. compactly ϵ-provisioned with the same parameter ϵ).*

[5]Although the translation of an SPJU query to a UCQ may incur an exponential size blowup [6], in this paper, query (and schema) size are assumed to be constant. In fact, in practice, SQL queries often present with unions already at top level.

Proof. Suppose Q_N can be compactly provisioned (the following proof also works when Q_N can be compactly ϵ-provisioned). Let b be the maximum size of the conjunctive queries in Q_L. Given an input instance I and a set H of k hypotheticals, we define a new instance $\hat{I} = Q_L(I)$ and a set $\hat{H}$ of $O(k^b)$ new hypotheticals as follows. For each subset $S \subseteq [k]$ of size at most b (i.e., $|S| \leq b$), define a hypothetical $\hat{h}_S(\hat{I}) = Q_L(I|_S)$ (though S is not a number, we still use it as an index to refer to the hypothetical $\hat{h}_S$).

By our definition of the semantics of complex queries, the group-by operation partitions $\hat{I}$ and each $\hat{h}_S$ into $p = |\Pi_{\overline{A}}(\hat{I})|$ sets. We treat each group individually, and create a sketch for each of them. To simplify the notation, we still use $\hat{I}$ and $\hat{H}$ to denote respectively the portion of the new instance, and the portion of each new hypothetical that correspond to, without loss of generality, the first group. In the following, we show that a compact provisioning scheme for Q_N with input $\hat{I}$ and $\hat{H}$ can be adapted to compactly provision $\langle Q_L; G_{\overline{A}}; Q_N \rangle$ for the first group. Since the number of groups p is assumed to be $\text{poly}(k, \log n)$, the overall sketch size is still $\text{poly}(k, \log n)$, hence achieving compact provisioning for the complex query.

Create a sketch for Q_N with input $\hat{I}$ and $\hat{H}$. For any scenario $S \in [k]$ (over H), we can answer the numerical query Q_N using the scenario $\hat{S}$ (over $\hat{H}$) where $\hat{S} = \{S' \mid S' \subseteq S \ \&\ |S'| \leq b\}$. To see this, we only need to show that the input to Q_N remains the same, i.e., $Q_L(I|_S)$ is equal to $\hat{I}|_{\hat{S}}$. Each tuple t in $Q_L(I|_S)$ can be derived using (at most) b hypotheticals. Since any subset of S with at most b hypotheticals belongs to $\hat{S}$, the tuple t belongs to $\hat{I}|_{\hat{S}}$. On the other hand, each tuple t' in $\hat{I}|_{\hat{S}}$ belongs to some $\hat{h}_{S'}$ where $S' \in S$, and hence, by definition of $\hat{h}_{S'}$, the tuple t is also in $Q_L(I|_S)$. Hence, $Q_L(I|_S) = \hat{I}|_{\hat{S}}$.

Consequently, any compact provisioning scheme for Q_N can be adapted to a compact provisioning scheme for the query $\langle Q_L; G_{\overline{A}}; Q_N \rangle$. □

Theorem 6.6 further motivates our results in Section 6.4 for numerical queries as they can be extended to these quite practical queries. Additionally, as an immediate corollary of

the proof of Theorem 6.6, we obtain that any boolean UCQ (i.e., any UCQ that outputs a boolean answer rather than a set of tuples) can be compactly provisioned.

Corollary 6.7. *Any boolean UCQ can be compactly provisioned using sketches of size $O(k^b)$ bits, where b is the maximum size of each CQ.*

Remark 6.5.1. *Deutch et al. [41] introduced query provisioning from a practical perspective and proposed* boolean provenance *[72, 57, 56, 105] as a way of building sketches. This technique can also be used for compactly provisioning boolean UCQs.*

Proof Sketch. Given a query Q, an instance I and a set H of hypotheticals we compute a small sketch in the form of a boolean provenance expression.

Suppose that each tuple of an instance I is annotated with a distinct provenance token. The provenance annotation of the answer to a UCQ is a monotone DNF formula Δ whose variables are these tokens. Crucially, each term of Δ has fewer than b tokens where b is the size of the largest CQ in Q. Associate a boolean variable x_i with each hypothetical $h_i \in H,\ i \in [k]$. Substitute in Δ each token annotating a tuple t with the disjunction of the x_i's such that $t \in h_i(I)$ and with false otherwise. The result Δ' is a DNF on the variables $x_1, \ldots, x_k$ such that each of its terms has at most b variables; hence the size of Δ' is $O(k^b)$. Now Δ' can be used as a sketch if the extraction algorithm sets to true the variables corresponding to the hypotheticals in the scenario and to false the other variables. □

Remark 6.5.2. *A similar approach based on rewriting boolean provenance annotations, which are now general DNFs, can be used to provision UCQ$^\neg$s under disjoint hypotheticals. The disjointness assumption insures that negation is applied only to single variables and the resulting DNF has size $O((2k+1)^b) = O(k^b)$.*

We further point out that the exponential dependence of the sketch size on the query size (implicit) in Theorem 6.6 and Corollary 6.7 cannot be avoided even for CQs.

Theorem 6.8. *There exists a boolean conjunctive query Q of size b such that provisioning Q requires sketches of size* $\min(\Omega(k^{b-1}), \Omega(n))$ *bits.*

Proof of Theorem 6.8. Consider the following boolean conjunctive query $Q_{\texttt{EXP}}$ defined over a schema with a unary relation A and a $(b-1)$-ary relation B.

$$Q_{\texttt{EXP}} \equiv ans() \ :- \ A(x_1), A(x_2), \ldots, A(x_{b-1}),$$
$$B(x_1, x_2, \ldots, x_{b-1})$$

We show how to encode a bit-string of length $N := \binom{k-1}{b-1}$ into a database I with $n = \Theta(N)$ tuples and a set of k hypotheticals such that given provisioned sketch of $Q_{\texttt{EXP}}$, one can recover any bit of this string with constant probability. Standard information-theoretic arguments then imply that the sketch size must have $\Omega(N) = \Omega(k^{b-1}) = \Omega(n)$ bits.

Let $(S_1, \ldots, S_N)$ be a list of all subsets of $[k-1]$ of size $b-1$. For any vector $\mathbf{v} \in \{0,1\}^N$, define the instance $I_{\mathbf{v}} = \{A(x)\}_{x \in [k-1]} \cup \{B(S_y)\}_{v_y = 1}$ (this is slightly abusing the notation: $B(S_y) = B(x_1, \ldots, x_{b-1})$ where $\{x_1, \ldots, x_{b-1}\} = S_y$), and a set $\{h_i\}_{i \in [k]}$ of k hypotheticals, where for any $i \in [k-1]$, $h_i(I) = \{A(i)\}$ and $h_k(I) = \{B(S_y)\}_{v_y = 1}$. To compute the i-th entry of $\mathbf{v}$, we can extract the answer to the scenario $S_i \cup \{k\}$ from the sketch and output 1 iff the answer of the query is true.

To see the correctness, $v_i = 1$ iff $B(S_i) \in h_k(I)$ iff $Q_{\texttt{EXP}}(I|_{S_i \cup \{k\}})$ is true. □

6.6. Comparison With a Distributed Computation Model

Query provisioning bears some resemblance to distributed computation: k sites want to jointly compute a function, where each site holds only a portion of the input. The function is computed by a *coordinator*, who receives/sends data from/to each site. The goal is to design protocols with small amount of communication between the sites and the coordinator (see [36, 37, 109, 32, 112, 29], and references therein). In what follows, we highlight some key similarities and differences between this distributed computation model and our query provisioning framework.

In principle, any protocol where data are only sent from the sites to the coordinator (i.e.,

one-way communication), can be adapted into a provisioning scheme with a sketch of size proportional to the size of the transcript of the protocol. To see this, view each hypothetical as a site and record the transcripts as the sketch. Given a scenario S, the extraction algorithm acts as the coordinator and computes the function from the transcripts of the hypotheticals in S. The protocol for counting the number of distinct elements in the distributed model introduced in [36] is an example (which in fact also uses the streaming algorithm from [21] as we do in Lemma 6.4.2).

For distributed protocols with two-way communication (e.g., [111, 32]), in general, there is no oblivious way to simulate them in the query provisioning framework. In fact, as we will show in the following, the power of the distributed computation model with two-way communication is *incomparable* to the query provisioning framework. Specifically, for k sites/hypotheticals (even when the input is partitioned, i.e., no overlaps), there are problems where a protocol with poly(k)-bit transcripts exists but provisioning requires $2^{\Omega(k)}$-bit sketches. Conversely, there are problems where provisioning can be done using poly(k)-bit sketches but any distributed protocol requires $2^{\Omega(k)}$-bit transcripts. We make these observations precise below.

Another source of difference between our techniques and the ones used in the distributed computation model is the typical assumption in the latter that the input is *partitioned* among the sites in the distributed computation model (i.e. no overlaps; see, for instance, [111, 37]). This assumption is in contrast to the hypotheticals with unrestricted overlap that we handle in our query provisioning framework. A notable exception to the no-overlaps assumption is the study of quantiles (along with various other aggregates) by [32]. However, as stated by the authors, they benefit from the coordinator sharing a summary of the whole data distribution to individual sites, which requires a back and forth communication between the sites and coordinator. As such the results of [32] can not be directly translated into a provisioning scheme. We also point out that the approximation guarantee we obtain for the quantiles query is stronger than the result of [32] (i.e. a relative vs additive approximation).

Finally, we use two examples to establish an exponential separation between the minimum sketch size in the provisioning model and the minimum transcript length in the distributed computation model, proving a formal separation between the two models.

Before describing the examples, we should note that, in the following, we assume that input tuples are made distinct by adding another column which contains *unique identifiers*, but the problems and the operations will only be defined over the part of the tuples without the identifiers. For instance, 'two sets of tuples are disjoint' means '*after removing the identifiers of the tuples,* the two sets are disjoint'.

A problem with poly-size sketches and exponential-size transcripts. The following SetDisjointness problem is well-known to require transcripts of $\Omega(N)$ bits for any protocol [96, 20]: Alice is given a set S and Bob is given a set T both from the universe $[N]$, and they want to determine, with success probability at least $2/3$, whether S and T are disjoint. Similarly, if each of the k sites is given a set and they want to determine whether all their sets are disjoint, the size of the transcript is also lower bounded by $\Omega(N)$ bits. If we let $N = 2^k$, the distributed model requires $\Omega(2^k)$ bits of communication for solving this problem.

However, to provision the SetDisjointness query (problem), we only need to record the pairs of hypotheticals whose sets are not disjoint (hence $O(k^2)$ bits). The observation is that if a collection of sets are not disjoint, there must exist two sets that are also not disjoint, which can be detected by recording the non-disjoint hypothetical pairs.

A problem with exponential-size sketches and poly-size transcripts. Consider a relational schema with two unary relations A and B. Given an instance I, we let $a = \sum_{A(x)\in I} 2^x$; then, the problem is to determine whether $B(a) \in I$ or not. Intuitively, A 'encodes' the binary representation of a value a, and the problem is to determine whether a belongs to 'the set of values in B'.

Let $n = 2^k$, and $I = \left\{A(x)_{x\in[k]}\right\} \cup \left\{B(y)\right\}_{y\in[n]}$. It is not hard to show that provisioning

this problem requires a sketch of size $2^{\Omega(k)}$, even when the input instance is guaranteed to be a subset of I (i.e., the largest value of A is k).

However, in the distributed model, there is a protocol using a transcript of size poly(k) bits: every site sends its tuple in A to the coordinator; the coordinator computes $a = \sum_{A(x)} 2^x$ and sends a to every site; a site response 1 iff it contains the tuple $B(a)$, and 0 otherwise.

6.7. Conclusions and Future Work

In this chapter, we initiated a formal framework to study compact provisioning schemes for relational algebra queries, statistics/analytics including quantiles and linear regression, and complex queries. We considered provisioning for exact as well as approximate answers, and established upper and lower bounds on the sizes of the provisioning sketches.

The queries in our study include quantiles and linear regression queries from the list of in-database analytics highlighted in [70]. This is only a first step and the study of provisioning for other core analytics problems, such as variance computation, k-means clustering, logistic regression, and support vector machines is of interest.

Another direction for future research is the study of queries in which *numerical* computations follow each other (e.g., when the linear regression training data is itself the result of aggregations). Yet another direction for future research is an extension of our model to allow other kinds of hypotheticals/scenarios as discussed in [41] that are also of practical interest. For example, an alternative natural interpretation of hypotheticals is that they represent tuples to be *deleted* rather than retained. Hence the application of a scenario $S \subseteq [k]$ to I becomes $I|_S = I \setminus (\bigcup_{i \in S} h_i(I))$. Using our lower bound techniques, one can easily show that even simple queries like count or sum cannot be approximated to within any multiplicative factor under this definition. Nevertheless, it will be interesting to identify query classes that admit compact provisioning in the delete model or alternative natural models.

BIBLIOGRAPHY

[1] Greenplum DB (Pivotal). http://pivotal.io/big-data/pivotal-greenplum-database.

[2] The MADlib Project. http://madlib.net.

[3] Google processing 20,000 terabytes a day, and growing. https://techcrunch.com/2008/01/09/google-processing-20000-terabytes-a-day-and-growing/, 2008.

[4] The data explosion in 2014 minute by minute - infographic. https://aci.info/2014/07/12/the-data-explosion-in-2014-minute-by-minute-infographic, 2014.

[5] Surprising facts and stats about the big data industry. http://cloudtweaks.com/2015/03/surprising-facts-and-stats-about-the-big-data-industry, 2015.

[6] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[7] Marek Adamczyk. Improved analysis of the greedy algorithm for stochastic matching. *Inf. Process. Lett.*, 111(15):731–737, 2011.

[8] Mohammad Akbarpour, Shengwu Li, and Shayan Oveis Gharan. Dynamic matching market design. In *ACM Conference on Economics and Computation, EC '14, Stanford , CA, USA, June 8-12, 2014*, page 355, 2014.

[9] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *STOC*, pages 20–29. ACM, 1996.

[10] Noga Alon, Noam Nisan, Ran Raz, and Omri Weinstein. Welfare maximization with limited interaction. *CoRR, abs/1504.01780. To appear in FOCS 2015*, 2015.

[11] Ross Anderson, Itai Ashlagi, David Gamarnik, and Yash Kanoria. A dynamic model of barter exchange. In *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015*, pages 1925–1933, 2015.

[12] Ross Anderson, Itai Ashlagi, David Gamarnik, and Alvin E. Roth. Finding long chains in kidney exchange using the traveling salesman problem. *Proceedings of the National Academy of Sciences*, 112(3):663–668, 2015.

[13] Sepehr Assadi, Sanjeev Khanna, and Yang Li. The stochastic matching problem with (very) few queries. In *Proceedings of the 2016 ACM Conference on Economics and Computation, EC '16, Maastricht, The Netherlands, July 24-28, 2016*, pages 43–60, 2016.

[14] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Val Tannen. Algorithms for provisioning queries and analytics. In *19th International Conference on Database Theory, ICDT 2016, Bordeaux, France, March 15-18, 2016*, pages 18:1–18:18, 2016.

[15] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Rakesh V. Vohra. Fast convergence in the double oral auction. In *Web and Internet Economics - 11th International Conference, WINE 2015, Amsterdam, The Netherlands, December 9-12, 2015, Proceedings*, pages 60–73, 2015.

[16] Sepehr Assadi, Sanjeev Khanna, Yang Li, and Grigory 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.

[17] Pranjal Awasthi and Tuomas Sandholm. Online stochastic optimization in the large: Application to kidney exchange. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, 2009*, pages 405–411, 2009.

[18] Andrey Balmin, Thanos Papadimitriou, and Yannis Papakonstantinou. Hypothetical queries in an OLAP environment. In *VLDB*, pages 220–231, 2000.

[19] Nikhil Bansal, Anupam Gupta, Jian Li, Julián Mestre, Viswanath Nagarajan, and Atri Rudra. When LP is the cure for your matching woes: Improved bounds for stochastic matchings. *Algorithmica*, 63(4):733–762, 2012.

[20] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, and D Sivakumar. An information statistics approach to data stream and communication complexity. In *FOCS*, pages 209–218. IEEE, 2002.

[21] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *RANDOM*. Springer, 2002.

[22] Dimitri Bertsekas. A distributed algorithm for the assignment problem. *Lab. for Information and Decision Systems, Working Paper, M.I.T., Cambridge, MA*, 1979.

[23] Dimitri P Bertsekas. *Linear network optimization: algorithms and codes*. MIT Press, 1991.

[24] Dimitri P. Bertsekas and David A. Castaon. A forward/reverse auction algorithm for asymmetric assignment problems. *Computational Optimization and Applications*, 1(3):277–297, 1992.

[25] Avrim Blum, John P. Dickerson, Nika Haghtalab, Ariel D. Procaccia, Tuomas Sandholm, and Ankit Sharma. Ignorance is almost bliss: Near-optimal stochastic matching with few queries. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation, EC '15, Portland, OR, USA, June 15-19, 2015*, pages 325–342, 2015.

[26] Avrim Blum, Anupam Gupta, Ariel D. Procaccia, and Ankit Sharma. Harnessing the power of two crossmatches. In *ACM Conference on Electronic Commerce, EC '13, Philadelphia, PA, USA, June 16-20, 2013*, pages 123–140, 2013.

[27] Matthew Caesar and Jennifer Rexford. BGP routing policies in ISP networks. *IEEE Network*, 19(6):5–11, Nov/Dec 2005.

[28] Edward H Chamberlin. An experimental imperfect market. *The Journal of Political Economy*, 56(2):95–108, 1948.

[29] Di Chen and Qin Zhang. Streaming algorithms for robust distinct elements. In *Proceedings of the 2016 International Conference on Management of Data, SIGMOD Conference 2016, San Francisco, CA, USA, June 26 - July 01, 2016*, pages 1433–1447, 2016.

[30] Ning Chen, Nicole Immorlica, Anna R. Karlin, Mohammad Mahdian, and Atri Rudra. Approximating matches made in heaven. In *Automata, Languages and Programming, 36th International Colloquium, ICALP 2009, Part I*, pages 266–278, 2009.

[31] Rajesh Chitnis, Graham Cormode, Hossein Esfandiari, MohammadTaghi Hajiaghayi, Andrew McGregor, Morteza Monemizadeh, and Sofya 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*, pages 1326–1344, 2016.

[32] G. Cormode, S. Muthukrishnan, and W. Zhuang. What's different: Distributed, continuous monitoring of duplicate-resilient aggregates on data streams. In *ICDE*, pages 20–31, 2006.

[33] Graham Cormode, Flip Korn, S Muthukrishnan, and Divesh Srivastava. Space-and time-efficient deterministic algorithms for biased quantiles over data streams. In *PODS*, pages 263–272. ACM, 2006.

[34] Graham Cormode and S Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1), 2005.

[35] Graham Cormode and S. Muthukrishnan. Space efficient mining of multigraph streams. In *PODS*, pages 271–282, 2005.

[36] Graham Cormode, S Muthukrishnan, and Ke Yi. Algorithms for distributed functional monitoring. *ACM Transactions on Algorithms (TALG)*, 7(2):21, 2011.

[37] Graham Cormode, S Muthukrishnan, Ke Yi, and Qin Zhang. Optimal sampling from distributed streams. In *PODS*, pages 77–86. ACM, 2010.

[38] Kevin P. Costello, Prasad Tetali, and Pushkar Tripathi. Stochastic matching with commitment. In *Automata, Languages, and Programming - 39th International Colloquium, ICALP 2012, Proceedings, Part I*, pages 822–833, 2012.

[39] Vincent P Crawford and Elsie Marie Knoer. Job matching with heterogeneous firms and workers. *Econometrica: Journal of the Econometric Society*, 1981.

[40] Gabrielle Demange, David Gale, and Marilda Sotomayor. Multi-item auctions. *The Journal of Political Economy*, pages 863–872, 1986.

[41] Daniel Deutch, Zachary G Ives, Tova Milo, and Val Tannen. Caravan: Provisioning for what-if analysis. In *CIDR*, 2013.

[42] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Dynamic matching via weighted myopia with application to kidney exchange. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence.*, 2012.

[43] John P. Dickerson, Ariel D. Procaccia, and Tuomas Sandholm. Failure-aware kidney exchange. In *ACM Conference on Electronic Commerce, EC '13.*, pages 323–340, 2013.

[44] John P. Dickerson and Tuomas Sandholm. Futurematch: Combining human value judgments and machine learning to match in dynamic environments. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 622–628, 2015.

[45] Shahar Dobzinski, Noam Nisan, and Sigal Oren. Economic efficiency requires interaction. In *STOC*, 2014.

[46] Alex Fabrikant and Christos H. Papadimitriou. The complexity of game dynamics: BGP oscillations, sink equilibria, and beyond. In *Proc. ACM SODA*, pages 844–853, 2008.

[47] Alex Fabrikant, Umar Syed, and Jennifer Rexford. There's something about MRAI: Timing diversity exponentially worsens BGP convergence. In *Proc. IEEE INFOCOM*, 2011.

[48] Eldar Fischer, Eric Lehman, Ilan Newman, Sofya Raskhodnikova, Ronitt Rubinfeld, and Alex 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.

[49] Philippe Flajolet and G Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of computer and system sciences*, 31(2):182–209, 1985.

[50] Daniel P Friedman and John Rust. *The double auction market: institutions, theories, and evidence*, volume 14. Westview Press, 1993.

[51] David Gale and Lloyd S Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, pages 9–15, 1962.

[52] Lixin Gao and Jennifer Rexford. Stable internet routing without global coordination. *IEEE/ACM Transactions on Networking*, pages 681–692, 2001.

[53] Shahram Ghandeharizadeh, Richard Hull, and Dean Jacobs. Heraclitus: Elevating deltas to be first-class citizens in a database programming language. *ACM Trans. Database Syst.*, 21(3):370–426, 1996.

[54] Anna C Gilbert, Yannis Kotidis, S Muthukrishnan, and Martin J Strauss. How to summarize the universe: Dynamic maintenance of quantiles. In *PVLDB*, pages 454–465. VLDB Endowment, 2002.

[55] Ashish Goel, Michael Kapralov, and Sanjeev Khanna. On the communication and streaming complexity of maximum bipartite matching. In *ACM-SIAM SODA*, 2012.

[56] T.J. Green. Containment of conjunctive queries on annotated relations. *Theory Comput. Syst.*, 49(2), 2011.

[57] T.J. Green, G. Karvounarakis, and V. Tannen. Provenance semirings. In *PODS*, pages 31–40, 2007.

[58] Michael Greenwald and Sanjeev Khanna. Space-efficient online computation of quantile summaries. In *ACM SIGMOD Record*, volume 30, pages 58–66. ACM, 2001.

[59] Michael B Greenwald and Sanjeev Khanna. Power-conserving computation of order-statistics over sensor networks. In *PODS*, pages 275–285. ACM, 2004.

[60] Timothy Griffin and Gordon T. Wilfong. An analysis of BGP convergence properties. In *SIGCOMM*, pages 277–288, 1999.

[61] Timothy G. Griffin and Geoff Huston. RFC 4264: BGP wedgies, 2005.

[62] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. Policy disputes in path vector protocols. In *IEEE ICNP*, 1999.

[63] Timothy G. Griffin, F. Bruce Shepherd, and Gordon Wilfong. The stable paths problem and interdomain routing. *IEEE/ACM Transactions on Networking*, 10(2):232–243, 2002.

[64] Timothy G. Griffin and Gordon Wilfong. A safe path vector protocol. In *Proc. IEEE INFOCOM*, 2000.

[65] Anupam Gupta and Viswanath Nagarajan. A stochastic probing problem with applications. In *Integer Programming and Combinatorial Optimization - 16th International Conference, IPCO 2013. Proceedings*, pages 205–216, 2013.

[66] Anupam Gupta and Francis X Zane. Counting inversions in lists. In *SODA*, pages 253–254. Society for Industrial and Applied Mathematics, 2003.

[67] Alexander J. T. Gurney, Limin Jia, Anduo Wang, and Boon Thau Loo. Partial specification of routing configurations. In *Workshop on Rigorous Protocol Engineering (WRiPE)*, 2011.

[68] Alexander J. T. Gurney, Sanjeev Khanna, and Yang Li. Rapid convergence versus policy expressiveness in interdomain routing. In *35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016, San Francisco, CA, USA, April 10-14, 2016*, pages 1–9, 2016.

[69] Venkatesan Guruswami and Krzysztof Onak. Superlinear lower bounds for multipass graph processing. In *CCC*, 2013.

[70] Joseph M. Hellerstein, Christopher Ré, Florian Schoppmann, Daisy Zhe Wang, Eugene Fratkin, Aleksander Gorajek, Kee Siong Ng, Caleb Welton, Xixuan Feng, Kun Li, and Arun Kumar. The madlib analytics library or MAD skills, the SQL. *PVLDB*, 5(12):1700–1711, 2012.

[71] Zengfeng Huang, Bozidar Radunovic, Milan Vojnovic, and Qin Zhang. Communication complexity of approximate matching in distributed graphs. In *STACS*, 2015.

[72] T. Imielinski and W. Lipski. Incomplete information in relational databases. *J. ACM*, 31(4), 1984.

[73] Daniel M Kane, Jelani Nelson, and David P Woodruff. An optimal algorithm for the distinct elements problem. In *PODS*, pages 41–52. ACM, 2010.

[74] Yashodhan Kanoria, Mohsen Bayati, Christian Borgs, Jennifer Chayes, and Andrea Montanari. Fast convergence of natural bargaining dynamics in exchange networks. In *SODA*, 2011.

[75] Howard J. Karloff. On the convergence time of a path-vector protocol. In J. Ian Munro, editor, *SODA*, pages 605–614. SIAM, 2004.

[76] Howard J. Karloff, Siddharth Suri, and Sergei 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.

[77] Raimondas Kiveris, Silvio Lattanzi, Vahab S. Mirrokni, Vibhor Rastogi, and Sergei 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.

[78] D.E. Knuth. *Mariages stables et leurs relations avec d&autres problèmes combinatoires:*. Collection de la Chaire Aisenstadt. Presses de l'Université de Montréal, 1976.

[79] Harold W. Kuhn. The hungarian method for the assignment problem. In *50 Years of Integer Programming 1958-2008 - From the Early Years to the State-of-the-Art*, pages 29–47. 2010.

[80] Craig Labowitz, Abha Ahuja, Abhijit Abose, and Farnam Jahanian. An experimental study of BGP convergence. In *ACM SIGCOMM*, 2000.

[81] Craig Labowitz, G. Robert Malan, and Farnam Jahanian. Internet routing instability. *IEEE/ACM Transactions on Networking*, 1998.

[82] Silvio Lattanzi, Benjamin Moseley, Siddharth Suri, and Sergei 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.

[83] Yong Liao, Lixin Gao, Roch A. Guerin, and Zhi-Li Zhang. Safe inter-domain routing under diverse commercial agreements. *IEEE/ACM Transactions on Networking*, 18(6):1829–1840, 2010.

[84] L. Lovász and D. Plummer. *Matching Theory*. AMS Chelsea Publishing Series. American Mathematical Soc., 2009.

[85] Gurmeet Singh Manku, Sridhar Rajagopalan, and Bruce G Lindsay. Approximate medians and other quantiles in one pass and with limited memory. In *ACM SIGMOD Record*, volume 27, pages 426–435. ACM, 1998.

[86] David F. Manlove and Gregg O'Malley. Paired and altruistic kidney donation in the UK: algorithms and experimentation. *ACM Journal of Experimental Algorithmics*, 19(1), 2014.

[87] Zhuoqing Morley Mao, Ramesh Govindan, George Varghese, and Randy H. Katz. Route flap damping exacerbates Internet routing convergence. In *Proc. ACM SIGCOMM*, 2002.

[88] D. McPherson, V. Gill, D. Walton, and A. Retana. RFC 3345: Border Gateway Protocol (BGP) persistent route oscillation condition, 2002.

[89] Vahab S. Mirrokni and Morteza 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.

[90] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Chapman & Hall/CRC, 2010.

[91] Heinrich H. Nax, Bary S. R. Pradelski, and H. Peyton Young. Decentralized dynamics to optimal and stable states in the assignment game. In *CDC*, 2013.

[92] Alessandro Panconesi and Aravind Srinivasan. Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds. *SIAM J. Comput.*, 26(2):350–368, 1997.

[93] Christos H. Papadimitriou. Algorithms, games, and the Internet. In *Proc. ACM STOC*, 2001.

[94] Cristel Pelsser, Olaf Maennel, Pradosh Mohapatra, Randy Bush, and Keyur Patel. Route flap damping made usable. In *PAM*, 2011.

[95] Bary S.R. Pradelski. Decentralized dynamics and fast convergence in the assignment game: Extended abstract. In *EC*, New York, NY, USA, 2015. ACM.

[96] Alexander A. Razborov. On the distributional complexity of disjointness. *Theor. Comput. Sci.*, 106(2):385–390, 1992.

[97] Yakov Rekhter, Tony Li, and Susan Hares. RFC 4271: A Border Gateway Protocol 4 (BGP-4), 2006.

[98] Alvin E Roth and John H Vande Vate. Random paths to stability in two-sided matching. *Econometrica: Journal of the Econometric Society*, 1990.

[99] Rahul Sami, Michael Schapira, and Aviv Zohar. Searching for stability in interdomain routing. In *Proc. IEEE INFOCOM*, pages 549–557, 2009.

[100] Michael Schapira, Yaping Zhu, and Jennifer Rexford. Putting BGP on the right path: the case for next-hop routing. In *ACM HotNets*, 2010.

[101] L.S. Shapley and M. Shubik. The assignment game i: The core. *International Journal of Game Theory*, 1(1):111–130, 1971.

[102] Vernon L Smith. An experimental study of competitive market behavior. *The Journal of Political Economy*, 70(2):111–137, 1962.

[103] Vernon L Smith. *Papers in experimental economics*. Cambridge University Press, 1991.

[104] João Sobrinho. Network routing with path vector protocols: Theory and applications. In *Proc. ACM SIGCOMM*, pages 49–60, 2003.

[105] D. Suciu, D. Olteanu, C. Ré, and C. Koch. *Probabilistic Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2011.

[106] Utku Unver. Dynamic kidney exchange. *Review of Economic Studies*, 77(1):372–414, 2010.

[107] Kannan Varadhan, Ramesh Govindan, and Deborah Estrin. Persistent route oscillations in interdomain routing. *Computer Networks*, 2000.

[108] Curtis Villamizar, Ravi Chandra, and Ramesh Govindan. RFC 2439: BGP route flap damping, 1998.

[109] David P Woodruff and Qin Zhang. Tight bounds for distributed functional monitoring. In *STOC*, pages 941–960. ACM, 2012.

[110] David P. Woodruff and Qin Zhang. When distributed computation is communication expensive. In *DISC*, 2013.

[111] Ke Yi and Qin Zhang. Optimal tracking of distributed heavy hitters and quantiles. *Algorithmica*, 65(1):206–223, 2013.

[112] Qin Zhang. Communication-efficient computation on distributed noisy datasets. In *Proceedings of the 27th ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2015, Portland, OR, USA, June 13-15, 2015*, pages 313–322, 2015.

[113] Mingchen Zhao, Wenchao Zhou, Alexander J. T. Gurney, Andreas Haeberlen, Micah Sherr, and Boon Thau Loo. Private and verifiable interdomain routing decisions. In *Proc. ACM SIGCOMM*, 2012.