Distributed Computation and Inference

by

Govind Ramnarayan

B.S., University of California Berkeley (2014)
S.M., Massachusetts Institute of Technology (2016)

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

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2020

Author . . . .
Department of Electrical Engineering and Computer Science
May 15, 2020

Certified by . . . .
Elchanan Mossel
Professor of Mathematics
Thesis Supervisor

Accepted by . . . .
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Distributed Computation and Inference

by

Govind Ramnarayan

Submitted to the Department of Electrical Engineering and Computer Science
on May 15, 2020, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

In this thesis, we explore questions in algorithms and inference on distributed data.

On the algorithmic side, we give a computationally efficient algorithm that allows parties to execute distributed computations in the presence of adversarial noise. This work falls into the framework of *interactive coding*, which is an extension of error correcting codes to interactive settings commonly found in theoretical computer science.

On the inference side, we model social and biological processes and how they generate data, and analyze the computational limits of inference on the resulting data. Our first result regards the reconstruction of *pedigrees*, or family histories, from genetic data. We are given strings of genetic data for many individuals, and want to reconstruct how they are related. We show how to do this when we assume that both inheritance and mating are governed by some simple stochastic processes. This builds on previous work that posed the problem without a "random mating" assumption.

Our second inference result concerns the problem of corruption detection on networks. In this problem, we have parties situated on a network that report on the identity of their neighbors - "truthful" or "corrupt." The goal is to understand which network structures are amenable to finding the true identities of the nodes. We study the problem of finding a single truthful node, give an efficient algorithm for finding such a node, and prove that optimally placing corrupt agents in the network is computationally hard.

For the final result in this thesis, we present a model of opinion polarization. We show that in our model, natural advertising campaigns, with the sole goal of selling a product or idea, provably lead to the polarization of opinions on various topics. We characterize optimal strategies for advertisers in a simple setting, and show that executing an optimal strategy requires solving an NP-hard inference problem in the worst case.

Thesis Supervisor: Elchanan Mossel
Title: Professor of Mathematics

Acknowledgments

The foremost person I need to thank is my adviser, Elchanan Mossel, who has been a reliable source of encouragement, support, and fascinating brainstorming and discussion. Despite the many demands on his time, he always managed to make time to meet with me when needed. I owe much of my development as a researcher and a thinker to him. Perhaps my favorite thing about working with Elchanan was how fun meetings with him were; meetings with him always had a good mix of interesting ideas and humor, and I hope that I will be able to replicate this experience for others in the future.

I would also like to thank Yael Kalai, who worked with me for years of my PhD and taught me to recognize and appreciate my growth during this time. Her work ethic and energy for learning new things inspire me. Additionally, none of my experiences in my PhD would have been possible if not for my adviser during my Master's thesis, Dana Moshkovitz. It was under her patient guidance that I gained the confidence in my abilities that made the research during my PhD possible. She is incredibly motivated and does fascinating research, and working with her has been a pleasure.

I would like to additionally acknowledge my thesis committee, Elchanan Mossel, Yael Kalai, and Ronitt Rubinfeld, for their prompt responses throughout the thesis process. In particular I must acknowledge Ronitt here, who agreed to join my committee with short notice and gave prompt responses when I needed them.

My work in graduate school would not have been possible without my amazing coauthors, including Gireeja Ranade, Anant Sahai, Dana Moshkovitz, Henry Yuen, Tom Gur, Ron Rothblum, Ran Gelles, Yael Kalai, Ran Canetti, Aloni Cohen, Nishanth Dikkala, Sarah Scheffler, Adam Smith, Yan Jin, Younhun Kim, Frederic Koehler, Ankur Moitra, Jan Hązła, Paxton Turner, and Elchanan Mossel. These collaborators taught me different ways of thinking and helped me better understand my sets of skills and interests within mathematics and computer science. I thank them all for their patience and brilliance as they worked with me, and hope they enjoyed working with me as much as I enjoyed working with them.

My interest in theoretical computer science started before my PhD, and I would be amiss to not thank the people most responsible for this. Anant Sahai and Gireeja Ranade oversaw my first major research project, and showed me how to turn collections of interesting observations and ideas into a cohesive research paper. Satish Rao and Prasad Raghavendra stoked my interest in complexity theory, and encouraged me to pursue graduate studies. Going back farther, my interest in theoretical computer science came from courses in probability and algorithms at UC Berkeley, in particular EE126 with Anant Sahai (Probability and Random Processes), CS170 (Efficient Algorithms and Intractable Problems) with Satish Rao, as well as late night conversations about math with Hood Chatham. In particular, EE126 introduced me to many other students whose curiosity and drive encouraged me to do research, and with whom I have kept in touch during my PhD, including Anuran Makur, Zayd Enam, Steve Yadlowsky, Diivanand Ramalingam, Sibi Venkatesan, Vikram Iyer, and Satej Soman.

Many of my most positive experiences in the Theory of Computation group came from informal interactions outside of research. I would like to thank Itay Berman, Akshay Degwekar, Saleet Mossel, Talya Eden, Sitan Chen, Tselil Schramm, Adam Sealfon, Dylan McKay, Rio LaVigne, Henry Yuen, Adam Bouland, Josh Alman, Andrea Lincoln, Nikhil Vyas, Christopher Musco, Cameron Musco, Maryam Aliakbarpour, Shyam Narayanan, Mohammad Bavarian, Shalev Ben-David, Robin Kothari, Pritish Kamath, Gautam Kamath, Nicole Wein, Alex Wein, Quanquan Liu, Slobodan Mitrović, Rati Gelashvilli, Justin Kopinsky, William Leiserson, Alex Lombardi, Dhiraj Holden, Gil Kur, Lisa Yang, Michael Coulombe, Luke Schaeffer, Daniel Grier, Heather Berlin, Ryan Williams, Virginia Williams, and Ronitt Rubinfeld for everything from fun conversations around the office and over lunch, nights of playing music and games, and discussions about life outside of research. I have also been lucky to have friends in the Math department, including Frederic Koehler, Younhun Kim, Paxton Turner, Jake Wellens, and Vishesh Jain. The memories of my interactions with people from the Theory of Computation group and the Mathematics department are some of the fondest I have had.

Particular thanks goes to my officemates in G628 and G575-A, including Gautam Kamath, Ilya Razenshteyn, Themis Gouleakis, Arturs Backurs, Sitan Chen, Nishanth Dikkala, Siddhartha Jayanti, Robin Hui, Mahi Shafiullah, Brynmor Chapman, Alex Makelov, Morris Yau, Sam Park, Tianren Liu, and Michael Cohen, for particularly memorable and animated conversations. In particular, I remember bizarre and playful hypothetical discussions with Michael from my first year, and miss them dearly.

Besides the community of students and faculty, another aspect that makes the Theory of Computation group so enjoyable is the amazing administrators, including Debbie Goodwin, Rebecca Yadegar, Joanne Hanley, Linda Lynch, and Patrice Macaluso. Their hard work and kindness enable the work of the entire group, and make the graduate school experience much more manageable. Debbie in particular often asked after my well-being during some of the more stressful times of my graduate school career, which I greatly appreciated. I also want to thank Janet Fischer and Alicia Duarte, who have patiently answered my questions through the PhD process.

I have been lucky to have had friends at MIT outside of the Theory group, including Hsin-Yu Lai, Igor Kadota, Nicha Apichitsopa, Tadayuki Yoshitake, Zied Ben Chaouch, Anuran Makur, Ganesh Ajjanagadde, James Noraky, Margaret Lee, Ty Christoff, Lily Yan, and Sara Sheffels. In particular, Sara has spent painstaking hours listening to my trials and tribulations, and taught me valuable lessons about maintaining personal relationships and separating work from the other parts of my life.

Finally, I would like to thank my family. My parents, Ravi and Vasantha, have been a constant source of encouragement during my PhD, and their love was a stabilizing force in my life when research seemed bleak. They have encouraged my passion for mathematics my entire life, and without them I never would have come to MIT or pursued a PhD.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In this thesis, we study questions in algorithms and inference that arise when data is distributed among many individuals. These types of questions have been widely studied in areas such as distributed algorithms and statistical inference. In distributed algorithms, researchers are interested in what computations can be done by the parties that have this distributed data. In statistics, researchers are interested in what a central party can learn when it sees data that comes from some generating process. We study both these types of questions in this thesis, focusing on what can be done by agents whose computations are limited to run in polynomial time, whether this be the parties in the setting of distributed algorithms, or the central party in the setting of inference.

1.1 Background and Outline of Thesis

On the algorithmic side, we analyze the ability of parties on a network to simulate a distributed computation in the presence of adversarial noise. Specifically, we suppose that multiple parties are situated on a network of communication links, and want to carry out some distributed, synchronous, protocol, where the protocol dictates to each party which symbols (if any) to send to their neighbors in each round. The protcocol can be *interactive*, meaning that the symbol sent by a party in a certain round may depend on which symbols it has received previously. The parties' goal is to compute some function of all of their input data, without having to communicate the (potentially very long) inputs themselves. The parties are hindered by an adversary that is permitted to insert and delete messages that are sent on any communication link throughout the protocol, with a restriction on the fraction of commmunication that can be inserted or deleted. Nevertheless, the parties want to carry out their protocol correctly, using the least amount redundancy possible to correct their errors.

The question we have described is known as the multiparty interactive coding problem with adversarial noise. To appreciate the motivation for our work, we briefly recap some of the relevant history of this problem. Interactive coding emerged from a rich study of how to protect transmissions from noise. The study of protecting one-way transmissions across a noisy channel was started by classical works of Shannon [148] and Hamming [73], and has been an extremely active field since (see [155] and [167] for two surveys within theoretical computer science). In the early 1990's, Schulman [144, 145] introduced and studied the problem of performing an *interactive* two-party computation over a noisy communication channel, where each party has private input data, and the errors on the channel can be adversarial or stochastic. As interactive proofs and computations are a core aspect of theoretical computer science (e.g. see Goldwasser, Micali, and Rackoff [65]), interactive coding extended the study of correcting errors in transmissions to settings that were increasingly interesting to computer scientists.

Subsequent work has greatly increased the scope of interactive coding to further settings that are interesting to computer scientists. Rajagopalan and Schulman extended this question to the multiparty setting, where n parties are situated with a network of communication links between them [139], and they addressed the case of stochastic noise. They considered protocols where every party sends messages on all its outgoing links in each round, and gave schemes for correcting errors for these protocols that did not significantly increase the round complexity of the underlying protocol. Multiparty interactive coding has received considerable attention recently [4, 57, 29, 79, 85, 2, 31].

The case of adversarial noise in the multiparty setting was left untouched for two decades, until the works of Jain, Kalai, and Lewko [85], and Hoza and Schulman [79]. Hoza and Schulman addressed the same protocols as [139], where every link carries a message in each round, and showed that these protocols can also be made resilient to a small fraction of adversarial noise.

In an attempt to broaden the scope of interactive coding, Jain, Kalai, and Lewko described a coding scheme for *arbitrary* protocols, where links in the protocol are permitted to carry no message. Their motivation was to make a coding scheme that applies to the types of computations that are often used in theoretical computer science. However, a drawback was that their scheme only applied to star graphs, and that it only tolerated adversarial corruptions, rather than insertions and deletions. Additionally, both [85] and [79] are currently computationally inefficient: they rely on the construction of a combinatorial object called a *tree code*, first defined by Schulman [144, 145] in the works that started the area of interactive coding. While the existence of sufficiently good tree codes dates back to Schulman's original work, there is currently no polynomial time algorithm for constructing asymptotically good tree codes, which these constructions rely on[1]. This brings us to the first question we address in this thesis:

Question 1. *Is it possible to have multiparty interactive coding against adversarial insertions and deletions, where the parties are* computationally efficient - *i.e. run in polynomial time in the size of the underlying graph and protocol?*

We answer this question in the affirmative, giving polynomial time algorithms for this setting. Our results work for arbitrary graphs, in the more general communication model of [85], where parties can opt to be silent in certain rounds.

There are other aspects of our result we find appealing. The model of communication in [85] was drawn from distributed algorithms and cryptography, where parties are permitted to be silent in the midst of a computation. However, the restriction of the

[1]An exciting line of recent work constructs tree codes over larger (polylogarithmic) alphabet size. See [34] for the construction of these new tree codes, and [123] for progress on decoding them.

protocol to a star graph is a major drawback in this sense; in both distributed computing and privacy, it is undesirable to have a single node (the center of the star) privy to all the communication in the network. So we see our extension of this model to all graphs as a significant bonus, as it enables interactive coding in this model for networks that do not have a central party. However, we lose slightly in parameters compared to [85] and [79]. We refer the reader to Chapter 2 for further details regarding our protocol and comparisons with previous work.

We note that the model of distributed algorithms we have described bears similarity to the rich area of *consensus* algorithms, in which a group of parties seek to agree on a certain value for the purposes of a computation. However, the noise model we consider is different. Algorithm designers for consensus often wish to make their algorithms tolerant to faulty nodes. By contrast, in multiparty interactive coding we look at the weaker noise model where the errors are only in transmissions: we assume that every node continues to operate correctly throughout the protocol, but that the adversary can tamper with some of the messages they send. This weaker noise model is necessary, as the types of protocols we are trying to protect are very brittle. Specifically, these protocols depend on the input of each and every party. If the adversary can simulate even one party, they can make the entire network believe that this party has a different input, thus changing the outcome of the protocol. We refer the reader to [102, 103, 131] for some well-known consensus algorithms in the setting of faulty nodes, to [124] for a survey of the applications of consensus to blockchain, and to [109] for distributed algorithms more broadly.

The rest of this thesis discusses centralized inference on data coming from multiple individuals. Recall that in distributed computation, we want the parties to solve computational problems without relying on a central party. However, in the setting of inference, we ask what a central observer can learn from seeing data generated by the different parties in the network. For example, social scientists may want to learn about human interaction by looking at data from social media, or a biologist may want to understand evolutionary history by observing present-day genetic data. In both of these situations, the relevant data is distributed among individuals, and is generated according to some process; a process governing social interactions in the former case, and a process of genetic mutations in the latter. The rest of this thesis focuses on these types of inference questions. Specifically, we study models of social and biological processes, and the information that a computationally bounded observer can glean from them.

In Chapter 3, we give an example from biology that falls into a classical paradigm of statistical learning theory, where we are interested in learning the latent parameters of some stochastic process. In this problem, individual genetic data is generated by a process of genetic inheritance from previous generations. Given data from individuals in the present day, we ask whether an observer can learn the familial relations that gave rise to this set of individuals. The process of genetic inheritance is assumed to follow a certain stochastic process, which is known to the observer. In particular, we want the observer to be able to compute the familial relations in polynomial time, while requiring as little data from each individual as possible. This is known as the *pedigree reconstruction* problem [159].

This problem has been widely considered in practice, with genetic testing services such as 23AndMe and Ancestry.com gaining significant traction as the cost of genetic sequencing has plummeted in recent years. However, most algorithms in the biology community tend to lack theoretical guarantees for the reconstruction (see Chapter 3 for discussion). Hence, researchers in mathematical biology initiated the mathematical study of pedigrees, and asked when pedigrees can be provably reconstructed [154, 159]. We build on this line of work in this thesis. The following question, asked by Steel and Hein in [154], forms the foundation of this line of questioning.

Question 2. *When can a pedigree be reconstructed exclusively from genetic data? In particular, what length of genetic sequences is necessary for the inference, and for how many generations in the past can relations be correctly constructed, as a function of the number of individuals?*

Steel and Hein [154] gave lower bounds on the amount of data required for pedigree reconstruction based on enumerating the number of pedigrees of a certain depth - for this result, they do not assume anything about the genetic inheritance process. They additionally proved that pedigrees can be reconstructed given pairwise ancestries of all the nodes. Follow-up work by Thatte and Steel [159] modeled the inheritance of symbols with various stochastic processes, and analyzed whether reconstruction was possible in these different settings.

In order to give an efficient algorithm for the central observer with as little data required as possible, we impose an additional modeling assumption on the problem. Namely, rather than simply modeling the way genetic sequences are inherited by an individual, we also model the mating habits of the underlying population with a stochastic process. Namely, we assume that the population is split into generations, that mating within each generation is monogamous and random, and that the number of children per couple is randomly sampled from a distribution with a fixed mean α.

These extra modeling assumptions make the task for the central observer tractable. Indeed, under these assumptions, we give a polynomial time algorithm for the observer to approximately reconstruct pedigrees far back into the past, with relatively few samples per individual. Our modeling assumptions and analysis bear similarity to tools used in other areas of mathematical biology, including population reconstruction [91, 92, 22], where the goal is to reconstruct human population migration and interbreeding, as well as phylogenetic reconstruction, where the goal is to reconstruct an evolutionary tree given gene samples [44, 114, 115, 116, 39]. We detail our reconstruction guarantees and these similarities in further detail in Chapter 3.

The remaining inference problems discussed in this thesis draw from the social sciences. In the pedigrees question, we concerned ourselves with the ability of a central observer to infer familial relations from genetic data. However, our observer was passive; she only wished to learn the true familial relations, but did not try to change the data in any way. However, questions stemming from social science often feature a central party wanting to make some sort of intervention on the data it sees. This phenomenon can be seen in previous computer science literature that has modeled the social sciences. In the viral marketing problem, also known as influence maximization [88], an advertiser wishes to target the most influential k nodes of some network, in order to optimally create a "word-of-mouth" campaign for their product. While the observer only needs to solve a static inference problem, they do this with the intent of influencing the network to buy their product. This suggests that we should consider two types of observers, with different motivations. The first one is passive, and wants to learn about the true state of the parties. The second one is active, and wants to change the data in some way to suit its own agenda.

In the pedigrees problem, we were interested in the number of symbols from each individual required for the observer to be able to infer the underlying state, and each symbol gave an independent signal of the underlying familial relations. In the following problems, we do not make any distributional assumption on the samples the observer sees. Each person provides their own opinion on some issues, and we do not consider it to be a "sample" from some larger distribution. Given this data, we want to know whether or not each observer can learn what they want to learn in polynomial time.

Following this general line of inquiry, we propose two models for processes inspired by social science. For the first model, we provide an efficient algorithm for inferring underlying states of these processes. For both processes, we classify the ability of computationally bounded agents to influence them.

In Chapter 4, we look at the problem of *corruption detection*, in which a central agency is trying to find a truthful party among a set of (possibly corrupt) parties. Truthful parties report the true affiliation of their neighbors (i.e. truthful or corrupt), while corrupt parties report arbitrarily. We classify the power of an adversary that wants to foil the central agency while corrupting the smallest number of parties.

This model dates back to the work of Preparata, Metze, and Chien [135], who studied it in the context of digital systems. They wanted to know which types of network structures enable the central agency to detect all the "corrupt nodes," which to them were faulty components of a digital system. Their work spawned many other works over the years (e.g. [87, 99, 166, 156, 37, 111, 72]). Alon, Mossel, and Pemantle [6] suggested using this model for corruption detection. They wanted to understand the ability of the central agency to find the identities of *most* nodes in the presence of bounded numbers of corruptions, and, in contrast with [135], they showed that bounded degree graphs can suffice for this. They did not assume anything about the corruptions other than a bound on the total number.

Our starting point is the work of [6]. Just like them, we are interested in the ability of the central agency to achieve its goal (which for us, is finding a single truthful node) in polynomial time. However, we make an additional modeling assumption that gives the question a new twist. Specifically, we assume that the corruptions are placed by a computationally bounded adversary. This gives us a two player setup: on one hand, the (passive) central agency, which wants to find a genuinely truthful party, and the active adversary, that wants to foil the central agency. This leads us to the following question.

Question 3. *Is a computationally bounded adversary as effective at foiling the central agency as an unbounded adversary? Namely, is there an efficient algorithm for the adversary to find the smallest set of nodes that prevents the central agency from finding a truthful node, on any graph?*

We find that, assuming a certain conjecture from hardness of approximation, the answer is no, in a very strong sense: the adversary can even be given a budget to corrupt hundreds of times more nodes than he needs, and still he cannot find a good set of nodes to corrupt within this budget. However, the central agency's task of finding a truthful node given reports turns out to be much easier. We formalize this observation in Chapter 4.

Finally, in Chapter 5, we propose a new mathematical problem aimed at modeling opinion polarization in society. In this model, an advertiser views the opinions of each member of the population, represented as high dimensional vectors, and chooses a campaign strategy by which to sell a product or idea. Once they have chosen a strategy, they broadcast their advertisements to the entire population. Their goal is to make the largest number of opinion vectors in the population align with its target opinion after some prescribed amount of time. For example, if the advertiser is selling detergent, their goal would be to make as many people have very positive opinions of their detergent brand as possible.

We study various strategies that the advertiser can take, and the effect that these strategies have on the opinion structure of society. The gist of the results is that, in our model for opinion updates, these strategies further polarize opinions.

From a theoretical computer science perspective, perhaps our most interesting result is in the case where one advertiser is trying to gain support in the long term; namely, their objective is to maximize their support asymptotically. We provide a simple, provably optimal strategy for the advertiser, and observe that this strategy clusters opinions in society around two clusters; one which loves the advertiser's product and one which hates it. Furthermore, we show that to achieve any optimal strategy, the advertiser must be able to learn halfspaces with noise, which is known to be NP-hard to approximate in high dimensions in the worst case [18, 68].

1.1.1 Organization of Thesis

The results in Chapter 2 are based on two papers that have been modified and rearranged to fit more naturally in one chapter. Section 2.2 and Section 2.4 are based on "Efficient Multiparty Interactive Coding, Part I: Oblivious Insertions, Deletions and Substitutions" [58], which was submitted to IEEE Transactions on Information Theory: Special Issue In Memory of Vladimir I. Levenshtein. Section 2.3 and Section 2.5 are based on "Efficient Multiparty Interactive Coding, Part II: Non-Oblivious Noise" [59], and was submitted to IEEE Transactions on Information Theory. Both papers were coauthored with Ran Gelles and Yael Kalai. A preliminary version of these two papers was submitted as a single paper to PODC 2019, titled "Efficient Multiparty Interactive Coding for Insertions, Deletions, and Substitutions" [60].

Chapter 3 is based on the paper "Efficient Reconstruction of Stochastic Pedigrees," which is joint work with Younhun Kim, Paxton Turner, and Elchanan Mossel. The preprint can be found on arXiv [93].

Chapter 4 is based on the paper "Being Corrupt Requires Being Clever, But Detecting Corruption Doesn't", which is joint work with Yan Jin and Elchanan Mossel. This appeared in ITCS 2019 [86].

Chapter 5 is based on the paper "A Geometric Model of Opinion Polarization", which is joint work with Jan Hązła, Yan Jin, and Elchanan Mossel. The preprint can be found on arXiv [74].

1.2 Common Preliminaries

We give some preliminaries that are common to the chapters in this thesis. Note that each chapter also has its own preliminaries, with notations and definitions that are specific to the work in that chapter.

For $n \in \mathbb{N}$ we denote by $[n]$ the set $\{1, 2, \ldots, n\}$. The $\log(\cdot)$ function is taken to base 2, while $\ln(\cdot)$ is base e.

For a distribution $\mathcal{D}$ we use $x \sim \mathcal{D}$ to denote that x is sampled according to the distribution $\mathcal{D}$.

$G = (V, E)$ denotes an undirected graph, where $n := |V|$ denotes the number of vertices and $m := |E|$ denotes the number of edges. For $v \in V$, let $\mathcal{N}(v)$ denote the neighborhood of v in G, i.e. $\mathcal{N}(v) = \{u : (u, v) \in E\}$.

A major tool we will use in multiple chapters is the famous Chernoff-Hoeffding bound.

Theorem 1.2.1. *Fix $p > 0$ and $\delta > 0$. Let $X = \sum_{i=1}^{N} X_i$ be a sum of independently identically distributed Bernoulli(p) random variables. Then*

$$\mathbb{P}[X \geq (p + \delta)N] \leq e^{-D(p+\delta\|p)N}$$

$$\mathbb{P}[X \geq (p - \delta)N] \leq e^{-D(p-\delta\|p)N}$$

where $D(x\|y) = x \ln\left(\frac{x}{y}\right) + (1-x)\ln\left(\frac{1-x}{1-y}\right)$ is the Kullback-Leibler divergence between two Bernoulli random variables with parameters x and y.

We will also use the following simpler forms of it:

$$\mathbb{P}(X > (1+\delta)pN) \leq \exp\left(-\frac{\delta^2}{3}pN\right)$$
$$\mathbb{P}(X < (1-\delta)pN) \leq \exp\left(-\frac{\delta^2}{2}pN\right)$$

Another fact we will use in the context of bounding probabilities is the following bound on binomial coefficients.

$$\left(\frac{n}{k}\right)^k \leq \binom{n}{k} \leq \left(\frac{ne}{k}\right)^k \tag{1.1}$$

Chapter 2

Efficient Multiparty Interactive Coding: Insertions, Deletions, and Substitutions

2.1 Introduction and Background

As mentioned in Chapter 1, we describe how to carry out distributed, interactive computations with noise. This work continues in a long line of works addressing communication on noisy channels. Communication channels may introduce noise of different types, for example, flipping transmitted bits. One notorious type of noise is *insertion and deletion* noise, that may add or remove bits from the transmissions due to synchronization mismatch [146]. The seminal work of Levenshtein [106] was the first to consider codes that correct insertions and deletions leading to a long line of research on correcting such errors, and bounding the capabilities of codes correcting such errors, e.g., [162, 157, 158, 107, 143, 40, 67, 142]. These codes apply to sending a transmission over a uni-directional channel.

In the early 90's, Schulman [144, 145] introduced and studied the problem of performing an *interactive* two-party computation over a noisy communication channel, rather than communicating in a uni-directional manner. Plenty of follow-up work appeared in the recent couple of decades extending Schulman's work in the two-party setting to a variety of noise and communication models (see [56] for a survey of these results). In [139], Rajagopalan and Schulman extended the two-party case and considered a network of n parties that wish to compute some function of their private inputs by communicating over an arbitrary[1] *noisy* network. The work of [139] shows that if each channel is the binary symmetric channel[2] (BSC_ε), then one can obtain a *coding scheme* that takes any protocol Π that assumes noiseless communication, and converts it into a resilient protocol that computes the same task over the noisy network.

The coding scheme in [139] defies noise by adding redundancy. The amount of added redundancy is usually measured with respect to the noiseless setting—the *rate* of the coding is the communication of the noiseless protocol divided by the communication of the noise-resilient one. The rate assumes values between zero and one, and ideally is bounded away from zero, commonly known as *constant* or *positive* rate. The rate may vary according to the network in consideration, for instance, the rate in [139] behaves as $1/O(\log(d+1))$ where d is the maximal degree in the network. Hence, for networks where the maximal degree is non-constant, the rate approaches zero as the network size increases.

The next major step for multiparty coding schemes was provided by Jain et al. [85] and by Hoza and Schulman [79]. In these works the noise is no longer assumed to be stochastic but instead is adversarial. That is, they consider worst-case noise where the only limit is the number of bits flipped by the adversary. They showed that as long as the adversary flips at most ε/m-fraction of the total communication, a coding scheme with a constant rate can be achieved, where ε is some small constant, and m is the number of communication links in the network.[3]

While both these works consider adversarial errors, they consider different communication models. [79] assumes that every party sends a single bit to all its neighbors in each round; this model is sometimes called *fully utilized*, and is commonly considered in

[1]By "arbitrary" we mean that the topology of the network can be an arbitrary graph $G = (V, E)$ where each node is a party and each edge is a communication channel connecting the parties associated with these nodes.

[2]That is, a channel that flips every bit with some constant probability $\varepsilon \in (0, 1/2)$.

[3][85] obtained this result for the specific star network, whereas [79] generalized this result to a network with arbitrary topology.

multiparty interactive coding [139, 79, 4, 29]. On the other hand, [85] uses a relaxed communication setting, in which parties may or may not speak in a given round; this setting is very common for distributed computations, and is relatively less studied in previous work on interactive coding. Furthermore, they do not assume that the *underlying* protocol is fully utilized, and simply use the (weaker) assumption that the order of speaking in the underlying protocol is known, and is independent of the parties' inputs. Naturally, one can convert any protocol in the non-fully-utilized model to a fully-utilized protocol by forcing all parties to speak at every round, and then apply an interactive coding scheme to the fully-utilized protocol. However, the conversion to a fully-utilized protocol may cause the communication complexity to increase by a factor of up to m, greatly harming the rate of the coding scheme.

2.1.1 Our Contributions

We design coding schemes for adversarial errors in the same communication model as [85]; namely, where parties may or not speak in a given round, and the underlying protocol is only assumed to have a fixed speaking order. One of the most interesting aspects of our work is that our coding schemes are *computationally efficient*, unlike either [85] or [79][4].

Also, we consider the stronger type of noise of *insertions and deletions* (albeit in the synchronous setting), where the noise may completely remove a transmission (so that the receiver is not aware that a bit was sent to him), or inject new transmissions (so that the receiver receives a bit while the sender didn't send anything). Insertion and deletion noise is more general, and is considered to be more difficult, than bit-flips. Indeed, a bit flip (a *substitution noise*) can be simulated by a deletion followed by an insertion. We note that insertions and deletions are trivially correctable in the fully utilized communication model; indeed, each party expects to hear a message from each of its neighbors in each round, so a deletion reduces to an erasure. By contrast, in the non-fully-utilized setting, insertions and deletions seem non-trivial to correct. Indeed, as first noted by Hoza [78],

[4]These coding schemes utilized a combinatorial object known a *tree code*, for which no efficient construction is known.

it seems *crucial* to allow insertion and deletion errors to make the problem non-trivial. To see this, consider a party that speaks once every two rounds—on an even round to communicate the bit '0' and on odd round to denote '1'. In fact, this communication is completely resilient to noise that only flips bits since only the timing of the transmission matters.

Finally, our coding schemes work on arbitrary topologies, in contrast to [85], which only applies to the star graphs. As mentioned above, [79] also handles arbitrary graphs, albeit in the fully utilized model.

In Section 2.2, we give an efficient interactive coding scheme with constant rate for arbitrary synchronous networks (not necessarily fully-utilized) that suffer from a certain fraction of insertion, deletion and substitution noise. In this part we assume that the parties pre-share a common random string (CRS), and assume the adversary is *oblivious* (i.e. their corruptions are fixed in advance, independent of the CRS, inputs, and communication).

We remark that, with $1/m$ noise rate, the adversary can completely corrupt a single link. Therefore, it is natural to allow the adversary to alter at most ε/m communication, as we do above.

In Section 2.3, we build on our result from Section 2.2. In our first coding scheme of Section 2.3 (Algorithm A), we show how to remove the shared randomness assumption. This scheme still assumes an oblivious adversary, whose corruptions are predetermined and are independent of the parties' randomness. In our second coding scheme of Section 2.3 (Algorithm B), we obtain a scheme that works even when the adversarial noise is non-oblivious, albeit, with slightly smaller noise resilience of $\varepsilon/m \log m$. Finally, if we remove only the assumption that the adversary is oblivious, but retain the assumption that the parties pre-share a long random string, then we obtain a scheme that is resilient to a higher level of noise, namely, to $\varepsilon/m \log\log m$-fraction of insertion and deletion errors (Algorithm C). We give details on this in Section 2.5. See Table 2.1 for a comparison of our new results (Algorithms A, B, and C) with the state of the art.

Related work As mentioned above, interactive coding was initiated by Schulman [144, 145]. Over the last several years there has been tremendous amount of work on interactive coding schemes in the two-party setting (e.g., [62, 27, 28, 24, 64, 55]), and in the multi-party setting (detailed below). We refer the reader to [56] for a survey on the field (and to references therein).

In what follows we only mention the schemes that are closely related to our setting, namely, ones that are either in the multiparty setting, or ones that are in the two-party setting but are resilient to insertions and deletions.

Coding schemes for insertions and deletions in the two-party setting were first constructed by Braverman, Gelles, Mao, and Ostrovsky [26]. As was noted above, in the model where in each round each party sends a single bit (which is the model used by most previous works, including [26]), insertions and deletions are only meaningful in the *asynchronous* model, as otherwise, such an error model is equivalent to the erasure model. Indeed, [26] considered the asynchronous model. We note that in the asynchronous model, a single deletion can cause a "deadlock", where both parties wait for the next incoming message. Therefore, Braverman et al. considered a model where any deletion is followed by an insertion, thus the protocol never "halts" due to noise. They note that the noise may delete a certain message and then inject a spoofed "reply" to the original sender. In this case, one party believes that the protocol has progressed by one step, while the other party is completely oblivious to this. This type of noise was called a *synchronization attack* as it brings the parties out of synch.

Braverman et al. [26] constructed a coding scheme for insertions and deletions in this model with constant communication rate, and resilience to constant fraction of noise. Later, Sherstov and Wu [150] showed that a very similar scheme can actually resist an *optimal* noise level. Both these schemes are computationally inefficient. Haeupler, Shahrasbi, and Vitercik [71] constructed an efficient scheme that is resilient to (a small) constant fraction of insertions and deletions. Furthermore, they constructed a scheme where the communication rate approaches 1 as the noise level approaches 0. Efremenko, Haramaty, and Kalai [43] considered the *synchronous* setting, where parties can send messages of arbitrary length in each round, and the adversary may insert and delete bits in the content of each message. They construct an efficient coding scheme with constant communication rate, and constant blowup in the round complexity, that is resilient to a small constant fraction of noise. All these works ([26, 150, 71, 43]) were in the two-party setting.

In the multiparty setting, Rajagopalan and Schulman [139] constructed a coding scheme for stochastic noise with rate $1/O(\log(d+1))$ for networks with maximal degree d. This implies a constant rate coding scheme for graphs with constant degree. Alon et al. [4] showed that if the topology is a clique, or a dense d-regular graph, then constant rate coding is also possible. Yet, Braverman, Efremenko, Gelles, and Haeupler [29] proved that a constant rate is impossible if the topology is a star. All the above works assume a synchronous fully-utilized network. Gelles and Kalai [57] showed that constant rate coding schemes are impossible also on graphs with constant degree, such as a cycle, assuming a synchronous, yet not fully-utilized model.

The case of adversarial noise in the multiparty setting was first considered by Jain, Kalai, and Lewko [85], who constructed a constant-rate coding scheme over a synchronous star network that is resilient to $O(1/n)$ fraction of noise. They did not assume that the network is fully utilized, and only assumed that the underlying noiseless protocol has a fixed speaking order. Hoza and Schulman [79] considered the fully utilized model with arbitrary topology and constructed a constant rate coding scheme that is resilient to $O(1/m)$ noise. Via routing and scheduling techniques, they show how to resist a fraction of $O(1/n)$-noise, while reducing the rate to $O(n/m\log(n))$. Both these schemes use tree-codes, and therefore are computationally inefficient.

Aggarwal, Dani, Hayes, and Saia [2] constructed an efficient synchronous coding scheme, assuming the parties use private point-to-point channels (i.e., with an oblivious adversary). They consider a variant of the fully-utilized model, where parties are allowed to remain silent, but the receiver will hear a constant bit set by the adversary for free (or further corrupted by the adversary as any other transmission). In their model the length of the protocol is not predetermined and may vary with the noise (similar to the two-party adaptive notion of [3]). Their coding scheme is resilient to an arbitrary (and *a priori* unknown) amount of bit-flips (as long as the noise-pattern is predetermined and independent of parties shared randomness), and has a rate of $O(1/\log(n\mathsf{CC}(\Pi)))$.

Censor-Hillel, Gelles, and Haeupler [31] constructed *asynchronous* coding schemes, where the parties do not know the topology of the network (an assumption that is very common in the distributed computation community). Their scheme is resilient to $O(1/n)$ noise and has a rate of $O(1/n\log^2 n)$, over an arbitrary topology.

Finally, we briefly mention some recent progress on constructing tree codes in polynomial time, as the explicit construction of efficient (that is, constant distance and constant rate) tree codes would make the conclusions of Jain, Kalai, and Lewko [85], as well as those of Hoza and Schulman [79], computationally efficient. Cohen, Haeupler, and Schulman [34] recently constructed tree codes with constant rate and distance over an alphabet of polylogarithmic size. They also note that this yields binary tree codes with rate $1/\log\log(n)$ (where n is the blocklength of the treecode), simply by translating the polylogarithmic alphabet into a binary string - this could be plugged into [85] and [79] with some loss in parameters if it could be decoded well. Narayanan and Weidner [123] give an exciting randomized polynomial time algorithm for decoding these codes based on the polynomial method, but this falls short of correcting up to constant distance. It would be interesting if these works can be pushed to get to the constant distance and constant rate tree codes that would make [85] and [79] computationally efficient.

2.1.2 Preliminaries

Notations and basic properties For a finite set Ω we let $\mathcal{U}_\Omega$ be the uniform distribution over Ω; we commonly omit Ω and write $x \sim \mathcal{U}$ when the domain is clear from context.

We note that our protocols in this chapter proceed in *iterations*, where in each iteration, a certain sequence of steps are repeated by each party. For any variable *var* that represents the state of some party in one of our algorithms (Algorithm 1, Algorithm A, Algorithm B, and Algorithm C), we let $var(i)$ denote the value of the variable var at the beginning of iteration i.

Multiparty interactive communication model We assume an undirected network $G = (V, E)$ of $n = |V|$ parties, $p_1, \ldots, p_n$, and $m = |E|$ edges, where p_i is connected to p_j if and only if $(p_i, p_j) \in E$. We identify parties with nodes, and treat p_i and node i as one. The network G is assumed to be a connected simple graph (i.e., without self-loops or multi-edges).

The communication model works in synchronous rounds as follows. At each round, any subset of parties may decide to speak. Each link is allowed to transmit at most one symbol per round in each direction from a constant-sized alphabet Σ. We will assume throughout this paper that $\Sigma = \{0, 1\}$ (both in the noiseless and noisy settings), however, our results extend to a larger alphabet as well. At each round, a party is allowed to send multiple (possibly different) symbols over multiple links. A *transmission* over a certain link at a certain round is the event of a party sending a message on this link at that round (if both parties send messages these are two separate transmissions).

We emphasize that, contrary to most previous work, our communication model is not fully-utilized and does not demand all parties to speak at each round on every communication channel connected to them; in fact we don't demand a certain party to speak at all at any given round.

Multiparty protocol Each party is given an input x_i, and its desire is to output $f_i(x_1, \ldots, x_n)$ for some predefined function f_i at the end of the process. A *protocol* π dictates to each party what is the next symbol to send over which channel (if any), as a function of the party's input, the round number, and all the communication that the party has observed so far. After a fixed and predetermined number of rounds, the protocol terminates and each party outputs a value as a function of its input and observed transcript. The *length of the protocol*, also called its *round complexity* $\mathsf{RC}(\pi)$ is the maximal number of rounds π takes to complete on any possible input. The communication complexity of the protocol (in bits), denoted by $\mathsf{CC}(\pi)$, is the total number of transmissions in the protocol times $\log|\Sigma|$. Since we assume $\Sigma = \{0,1\}$, the communication complexity equals the number of transmissions.

Throughout this chapter we denote the underlying (i.e., noiseless) interactive protocol that the parties are trying to compute by Π. We will usually use the notation $|\Pi|$ to denote the length of the (noiseless) protocol *in chunks* rather than in rounds; see Section 2.2.3 for details on partitioning protocols into chunks. We assume that the noiseless protocol has the property that the speaking order is independent of the inputs that the parties receive, and may depend only on the messages the party received.[5]

Noise model We concern ourselves with simulating multiparty interactive protocols Π among n parties over a *noisy* network. A single transmission over a noisy channel with alphabet Σ is defined as the function

$$\mathsf{Ch} : \Sigma \cup \{*\} \to \Sigma \cup \{*\},$$

where $*$ is a special symbol $* \notin \Sigma$ that means “no message”. Given a single utilization of the channel, we say the transmission is corrupt (or noisy) if $y = \mathsf{Ch}(x)$ and $y \neq x$. Specifically, if $x, y \in \Sigma$, this event is called a *substitution noise*, if $x = *$ and $y \in \Sigma$ the event is called an *insertion*, and if $x \in \Sigma$ and $y = *$ the event is called a *deletion*.

We stress that the noise may have a crucial effect on the protocol executed by the parties. Not only its correctness may be harmed, but also its length and communication may vary. Hence, in the noisy setting we redefine $\mathsf{RC}(\Pi)$ and $\mathsf{CC}(\Pi)$ to be the length and communication complexity, respectively, of a *given instance* of the protocol (which is determined by the inputs, the randomness, and the noise pattern). Moreover, we emphasize that, opposed to the fully utilized model where the number of rounds fixes the communication complexity, in our model these two are only related by the trivial bound $\mathsf{CC}(\Pi) \leq 2|E| \log |\Sigma| \cdot \mathsf{RC}(\Pi)$. We note that the gap between these two may be substantial.

The *fraction of noise* observed in a given instance, is the fraction of corrupt transmissions out of all the transmissions in that instance. For the binary case the noise fraction can be

[5]While we assume a fixed order of speaking for the noiseless protocol Π, we emphasize that this requirement will not apply on the coding scheme that simulates Π over the noisy network (similarly to [85]).

written as,

$$\mu = \frac{\#(\text{noisy transmissions})}{\mathsf{CC}(\Pi)}.$$

This is also known as *relative* noise.

An *oblivious* adversary is an adversary that pre-determines its noise attack, independently of the inputs and randomness of the parties. Our results in this work (the first part) apply to any oblivious adversary with a sufficiently limited number of errors. For concreteness, we consider two types of oblivious adversaries: *fixing* adversaries and *additive* adversaries.

Assuming a binary alphabet, an oblivious fixing adversary fixes a noise pattern $e = \{0,1,2,\perp\}^{2|E|\cdot\mathsf{RC}(\Pi)}$. In particular, the entry $e_{i,(u,v)} \in \{0,1,2,\perp\}$ determines whether noise occurs on the link $u \to v$ in round i and, if so, the new message on the link $u \to v$ in round i. Specifically, we interpret $\perp$ as leaving the communication as-is, 2 as fixing no message, and 0 and 1 as fixing 0 and 1 on the link respectively. The number of errors is the number of non-$\perp$ entries of e[6].

We also consider additive adversaries [21] (see, e.g., [36, 69, 63] for applications). An oblivious additive adversary fixes a noise pattern $e = \{0,1,2\}^{2|E|\cdot\mathsf{RC}(\Pi)}$ that defines the noise per each link in each round of the protocol. In particular, the entry $e_{i,(u,v)} \in \{0,1,2\}$ determines the noise added to the link $u \to v$ in round i. Assuming u transmits to v in iteration i the message $t \in \{0,1,2\}$ (where 2 denotes the case of no message, i.e., $*$), then v receives the transmission $t + e_{i,(u,v)} \mod 3$. The number of errors is the number of non-zero entries in e.

For both adversaries, the noise fraction is the number of errors divided by the actual communication given that error pattern (and specific inputs and randomness).

Remark 2.1.1. *We prove our results for both oblivious fixing and additive adversaries. As far as oblivious adversaries go, the fixing adversary seems more natural to us, and so for Section 2.2, we find the result most interesting with the fixing adversary. However, our end goal is to get*

[6]We note that this adversary is charged for an error whenever it tries to fix a link, even if the message sent on this link happens to be the message the adversary wanted to send.

a result for non-oblivious *adversaries, which we do in Section 2.3. To make our extension to non-oblivious adversaries in Section 2.3 simpler, we need to prove resilience against the oblivious* additive *adversary in this paper. We think the oblivious additive adversary is additionally interesting in its own right.*

Coding scheme—a noise-resilient protocol A coding scheme is a scheme that converts any protocol Π into a noise-resilient protocol $\tilde{\Pi}$ that simulates Π correctly with high probability on any given input. We say that a protocol $\tilde{\Pi}$ simulates Π correctly on a given input if each party can obtain its output corresponding to Π from the transcript it sees when executing $\tilde{\Pi}$. The protocol $\tilde{\Pi}$ is said to be resilient to μ-fraction of noise (with probability p), if it simulates Π correctly (with probability at least p) when executed over a noisy network with adversarial noise that corrupts at most μ-fraction of the transmissions in any instance of the protocol.

Our coding schemes will proceed in *iterations*. In each iteration, each party will perform a sequence of actions. Given an iteration i and an edge $(u,v) \in E$, we will denote the tuple (i,u,v) as a *triple* in this chapter. This is convenient notation in this chapter, and this meaning does not carry over to any other chapter (in particular we use the notation of "triples" to mean tuples of length 3 in Chapter 3).)

Hash functions We use an inner-product based hash function. The hash function is seeded with a random string s such that each bit of the output is an inner product between the input x and a certain part of s (using independent parts of s for different output bits). Formally,

Definition 2.1.2 (Inner Product Hash Function)**.** *The inner product hash function $h : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}^\tau$ is defined for any input x of length $|x| = L$ and seed s of length $|s| = \tau L$ (for $\tau \in \mathbb{N}$), as the concatenation of τ inner products between x and disjoint parts of s, namely,*

$$h(x,s) = \langle x, s[1,L]\rangle \circ \cdots \circ \langle x, s[(\tau-1)L+1, \tau L]\rangle.$$

We use the shorthand $h_s(x) \overset{\text{def}}{=} h(x,s)$.

We sometimes abuse notation and hash a ternary-string x (or a string over a larger alphabet). In this case, assume we first convert x into a binary string in the natural manner (each symbol separately, using $\lceil \log_2 3 \rceil = 2$ bits) and then hash the binary string. The seed length should increase appropriately (by at most a constant).

The following is a trivial property of this hash function, stating that, given a uniform seed, the output is also uniformly distributed.

Lemma 2.1.3. *For any* $\tau, L \in \mathbb{N}$, $x \in \{0,1\}^L \setminus \{0\}$, *and any* $r \in \{0,1\}^\tau$

$$\mathbb{P}_{s \sim \mathcal{U}}[h_s(x) = r] = 2^{-\tau}.$$

where $\mathcal{U}$ *is the uniform distribution over* $\{0,1\}^{\tau \cdot L}$.

It is easy to see Lemma 2.1.3 also implies that the collision probability of the inner product hash function with output length τ is exactly $2^{-\tau}$, since given two strings x and y such that $x \neq y$, the Lemma implies that the probability that $\mathbb{P}_{s \sim \mathcal{U}}[h_s(x-y) = \mathbf{0}] = 2^{-\tau}$.

2.2 Part I: Oblivious Noise

2.2.1 Introduction

In this first part of our chapter on interactive coding, we prove the following theorem. We recall that *oblivious* noise is predetermined, and in particular independent of the parties' randomness.

Theorem 2.2.1 (Coding for oblivious noise assuming shared randomness, informal)**.** *Let $G = (V, E)$ be an arbitrary synchronous network with $n = |V|$ nodes and $m = |E|$ links, and assume any two neighbours share a random string. For any noiseless protocol Π over G with a predetermined order of speaking, and for any sufficiently small constant ε, there exists an efficient coding scheme that simulates Π over a noisy network G. The simulated protocol is robust to adversarial insertion, deletion, and substitution noise, assuming at most ε/m-fraction of the communication is corrupted. The simulated protocol communicates $O(\mathsf{CC}(\Pi))$ bits, and succeeds with probability at least $1 - \exp(-\mathsf{CC}(\Pi)/m)$, assuming the noise is oblivious.*

In the next subsections we overview the key ideas required for performing interactive coding in the multiparty setting with constant overhead, and discuss related work. Our efficient coding scheme against an oblivious adversary assuming shared randomness is formally described in Section 2.2.3 and analyzed in Sections 2.2.4 - 2.2.6. The main sketch of the proof and the main technical contribution of this paper is contained in Section 2.2.4. Some parts of the coding scheme are based on the meeting point mechanism of [70] and are given in Section 2.4 for completeness.

2.2.2 Coding Scheme: Key Ideas

In this section we motivate the elements of our coding scheme at a high level. The basic idea towards constructing a multiparty coding scheme is to have each pair of parties perform a two-party coding scheme [139, 85, 79]. However, merely correcting errors in a pairwise manner is insufficient, since if a pair of parties found an inconsistency and backtracked, this may cause new inconsistencies (in particular, between these parties and their other neighbors). In [85], this problem was solved by assuming there is one party who is connected to all parties (i.e., the star topology). This party has a global view on the progress of the simulation in the *entire network*, and hence can function as the "conductor of the orchestra."

In our setting, no such central party exists and consequently no party knows the state of the simulation in the entire network. Instead, each party only sees its local neighborhood, and needs to propagate its "status" to the entire network in a completely decentralized manner.

We mention that Hoza and Schulman [79] also considered an arbitrary topology, but they consider the fully-utilized model. Correcting errors efficiently in the non-fully-utilized model seems to be trickier; we elaborate on this towards the end of this section.

In order to keep our simulation *efficient*, as opposed to previous works in the multi-party setting which used the (inefficient) tree-code approach, we use the rewind-if-error approach [144, 25, 96, 70, 55] (see also [56]). Namely, each two neighboring parties send a hash of their simulated pairwise transcripts, and if the hashes do not match, then an error is detected, and the two parties initiate a "meeting-points" mechanism [144, 70] in order to find a previous point where their transcripts agree.

Since we want a constant-rate coding scheme with error rate up to $\Omega(1/m)$, our goal is to ensure that any corruption only causes the network to waste $O(m)$ communication. Indeed, if the parties need K communication to correct a single error, then constant-rate coding implies a maximal noise level of $O(1/K)$. If the noise level is higher, the communication required to correct the errors is already too high to preserve a constant-rate coding. As a consequence of the above, in our coding schemes the parties will have to detect errors by communicating $O(1)$ bits per check, so that the consistency-checking costs $O(m)$ communication overall. In hindsight, this will restrict the parties to use hash functions with *constant-sized outputs*.

Let us now describe a preliminary attempt at a coding scheme. The coding scheme will have many *iterations*, where each iteration will consist of the following. First, every pair of parties will send each other hashes of their simulated pairwise transcripts. If these match, they continue simulating with each other for a small number of rounds; else, they use the meeting points mechanism to find a common "meeting" point where both their partial transcripts agree and rewind their partial transcript back to that point. Until they have done so, they refuse to simulate with any other party. The parties continue repeating these two steps (1. Send Hashes / Do Meeting Points and 2. Simulate / Stay Silent) until they finish simulating the underlying protocol. Each step will take a fixed number of rounds, so, since we are in the synchronous model, the parties will always be able to tell which step currently is being carried out.

This naive coding scheme has a clear but important flaw. Once a party u decides to rewind due to an error on the link (u,v), this has an effect on the simulation of u with its other neighbors. In order to ensure correctness of the overall simulation, u must rewind the simulation with each and every one of its neighbors that may be affected. It is tempting to rewind u's neighborhood by having u immediately truncate its simulated transcript for each adjacent link (u,w), to the same place it rewound the link (u,v). After such a truncation, a discrepancy would appear on the link (u,w) since w did not change its simulated transaction, and the meeting points mechanism would resolve any such resulting discrepancy in the next iteration. In particular, an error on the link (u,v) would cause u and v to perform meeting points and rewind their transcript to a consistent point. Then, in the next iteration, u will be inconsistent with its other neighbors and will initiate meeting points with all its other neighbors, resulting in them rewinding their transcripts accordingly. In the following iteration, any neighbor w of u would find an inconsistency with any other party in its *own* neighborhood, initiate meeting points with them, and so on until the entire network has rewound.

While the entire network eventually rewinds in this scheme, the number of iterations this takes is proportional to the diameter of the network. Recall that to ensure constant rate, we need to correct any error with at most $O(m)$ communication (with high probability), which in our case means just $O(1)$ iterations. So this naive approach could *not* achieve constant rate. Even worse, if magically we could augment the consistency check step so that news propagates through the entire network in just a single iteration, this approach would still fail with high probability. The consistency checking involves exchanging hashes of transcripts and verifying whether they match. *Hash collisions* are a major problem here; a hash collision will cause parties u and v to believe that their transcripts are consistent, when really they are not. Since we are restricted to use constant-sized hashes to keep the rate constant, hash collisions occur with constant probability in each exchange. Hence, in the course of the (at least) $n-1$ hash exchanges required to inform all n parties about the error, hash collisions occur with overwhelming probability. If so, with high probability many parties will not be informed of the error and will not rewind their transcript.

What saves us here is that u does not need to initiate a meeting points protocol to tell w to rewind—u can tell w to rewind *directly*. The meeting points protocol is designed to find a point where the transcripts of u and w are consistent, when neither u nor w has any idea where that point may be. But in the above case, u is confident that w should rewind, and should simply inform w it should rewind. To summarize the above ideas, an iteration of our coding scheme now consists of the following steps: 1. Consistency Check, 2. Simulate / Stay Silent, and 3. Rewind request (if necessary). As before, each of these phases takes a fixed number of rounds, so the parties are always in sync.

Some care needs to be taken with the Rewind phase. Specifically, u may not be able to instantly communicate to its neighbor w how far it wants w to rewind. For instance, u may have rewound a very large number of rounds, and even telling its neighbor how many rounds to rewind would require a large amount of communication; this cannot be communicated in the fixed number of rounds that the rewind phase contains. Instead, u should simply tell w to rewind a small amount at a time, spanning the rewind process over several consecutive iterations if needed. To summarize, the rewind phase allows the parties to gradually synchronize with each other without the use of unreliable hash functions, allowing the parties to return to simulation while circumventing the issues that hash collisions bring. This is one of the key conceptual observations behind our work.

There is one final issue that is specific to the non-fully-utilized model. Recall that in the non-fully-utilized model, communication may be sparse. For concreteness, consider the line network (i.e., a path graph, where party $i \in \{1, 2, \ldots, n-1\}$ is connected to party $i+1$). Consider the example protocol, where party 1 communicates back-and-forth with party 2 for n rounds while other parties remain silent, then party 2 communicates with party 3 for n rounds while other parties remain silent, and so on, until the communication reaches parties $n-1$ and n, and bounces back towards party 1. The communication that occurs in any set of n rounds of this underlying protocol is n, since only a single party is speaking at any given round. An iteration of our coding scheme would give the parties n rounds to simulate, after which they perform a single consistency check.

However, when *simulating* the underlying protocol in our coding scheme, the parties may not all be in agreement about which round they are simulating. For example, parties 1, 2, and 3 could think they are simulating round 1 (where parties 1 and 2 talk a lot), while party 4, 5, and 6 think they are simulating round $3n+1$ (where parties 4 and 5 talk a lot), and generally parties $3i+1, 3i+2$, and $3i+3$ think they are simulating round $i \cdot n+1$. In this case, there will be $\Omega(n^2)$ communication in just one iteration of our coding scheme (in $O(n)$ rounds). This is potentially disastrous. Not only might this lead to the communication blowup of the coding scheme being super-constant, but also the adversary would have the budget to place $\Omega(\varepsilon n)$ additional errors, and may be able to use these errors to derail any attempts of getting the simulation back on track.

To avoid this issue altogether, we introduce a "flag-passing" phase in which each party informs the entire *network* whether all seems correct and the simulation should continue, or if it sees an inconsistency and the network should idle while this is fixed. This fixes the issue above, since party 3 will realize that there is an inconsistency between its transcripts with parties 2 and 4, and will notify the network to avoid simulation.

Putting it all together, our resilient protocol consists of repeatedly executing the following four steps in order: (i) consistency check, (ii) flag passing, (iii) simulation, and

(iv) rewind. The coding scheme cycles through these four phases in a fixed manner, where each step takes a fixed amount of rounds to avoid ambiguity.

The formal specification of each phase, along with the formal coding scheme assuming oblivious noise and a common shared randomness is depicted as Algorithms 1–3 below (see Section 2.2.3 for full details and notations). The meeting points procedure is given in Algorithm 11, in Section 2.4.

2.2.3 Coding scheme for oblivious adversarial channels

Overview

The high-level description of the simulation is as follows. The basic mechanism is the rewind-if-error approach from previous works [144, 25, 70] (see also [56]). In particular, the parties execute the noiseless protocol Π for some rounds and then exchange some information to verify if there were any errors. If everything seems consistent, the simulation proceeds to the next part; otherwise, the parties rewind to a previous (hopefully consistent) point in Π and proceed from there.

Note that since multiple parties are involved, it may be that some parties believe the simulation so far is correct while others believe it is not. Yet, even if one party notices an inconsistency, the entire network may need to rewind. Hence, we need a mechanism that allows propagating the local view of each party to the entire network.

Our simulation algorithm consists of repeatedly executing the following four phases: (i) consistency check, (ii) flag passing, (iii) simulation, and (iv) rewind. The simulation protocol cycles through these four phases in a fixed manner, and each such cycle is referred to as an *iteration*. Each phase consists of a fixed number of rounds (independent of the parties' inputs and the content of the messages exchanged). Therefore, there is never an ambiguity as to which phase (and which iteration) is being executed. We next describe each phase (not in the order they are preformed in the protocol).

(i) **Simulation:** In this phase the parties simulate *a single chunk* of the protocol Π. Specifically, we split Π into chunks—consecutive sets of rounds—where at each chunk $5K$ bits are being communicated, for some $K \geq m$ that is fixed throughout the simulation and such that K is divisible by m. Jumping ahead, we note that K is set to be m in the protocol we construct in this section, which only considers to oblivious adversaries, and is set to $m \log m$ in Algorithm B in Section 2.3 which considers arbitrary (non-oblivious) adversaries. Note that since the speaking order in Π is fixed and predetermined, the partition into chunks is independent of the inputs and can be done in advance. We assume without loss of generality that each party speaks at least once in each chunk (this is without loss of generality since one can preprocess Π to achieve this property while increasing the communication complexity by only a constant factor).

In this phase, the parties "execute" the next chunk of Π, sending and receiving messages as dictated by the protocol Π.

This phase *always* takes $5K$ rounds, which is the maximal number of rounds required to simulate $5K$ transmissions of Π. It may be that the simulation of a specific chunk takes fewer rounds; in this case, the phase still takes $5K$ rounds where all the parties remain silent after the chunk's simulation has completed until $5K$ rounds have passed.

We note that some parties may be aware that the simulation so far contains errors that were not corrected yet (jumping ahead, this information can be obtained via local consistency checks that failed or via the global flag-passing phase, described below). When we reach the simulation phase, these parties will send a dummy message $\perp$ to their neighbors and remain silent for $5K$ rounds until the simulation phase completes.

(ii) **Consistency check:** The main purpose of this phase is to check whether each two *neighboring parties* $u, v \in V$ have consistent transcripts and can continue to simulate, or whether instead they need to correct prior errors. This phase is based on the *meeting points* mechanism [144, 70], which allows the parties to efficiently find the highest chunk number up to which they both agree.

Roughly speaking, every time the parties enter this phase, they exchange a hash of their current transcripts with each other. If the hashes agree, the parties believe that everything is consistent and effectively continue with simulating Π. If the hashes do not agree, the parties try to figure out the longest point in their transcript where they do agree. To this end, they send hashes of prefixes of their transcript until the hashes agree. In our setting, each time the parties enter the "consistency check" phase they perform a *single* iteration of the meeting-points mechanism [70], which consists of sending two hash values. If the hashes mismatch, they will send the next two hash values (of some prefixes of the transcript, as instructed by the meeting-points mechanism) *next time they enter the consistency check phase*.[7]

Note that the above is performed between each pair of adjacent parties, in parallel over the entire network.

(iii) **Flag passing:** In the flag passing phase, the parties attempt to synchronize whether or not they should continue the simulation of Π in the next simulation phase. As mentioned, it may be that some parties believe that the simulation so far is flawless while others may notice that there are some inconsistencies. In this phase the information about (2-party) inconsistencies is propagated to all the parties.

Roughly, if any party believes it shouldn't continue with the simulation, it notifies all its neighbors, which propagate the message to the rest of the network, and no party will simulate in the upcoming simulation phase. However, if all parties believe everything is consistent then no such message will be sent, and all the parties will continue simulating the next chunk of Π.

Technically speaking, the parties accomplish this synchronization step by passing a "flag" (i.e., a stop/continue bit) along a spanning tree $\mathcal{T}$ of G. Namely, each party receives flags from each of its children in $\mathcal{T}$. If one of the flags is stop, or if the party

[7]In addition to exchanging two hash values corresponding to prefixes of the transcript, the parties also exchange a hash indicating how long they have been running the meeting-points mechanism; see Section 2.4 for a full description.

sees inconsistency with one of its neighbors, it sends a stop flag to its parent in the tree. Otherwise, it sends its parent the continue flag. After the root of $\mathcal{T}$ receives all the flags, the root propagates the computed flag in the opposite direction back to the leafs. If there is no channel noise in this phase, it is clear that all parties are synchronized regarding whether the simulation should continue or not (recall that a party sends a dummy message during the simulation phase if its flag is set to stop).

(iv) **Rewind:** In the rewind phase, each party tries to correct any obvious (i.e., length-wise) inconsistencies with their neighbors. Recall that the meeting-points mechanism allows two neighboring parties to truncate their mis-matching transcripts to a prefix on which both parties agree. However, this may cause inconsistencies with all their other neighbors. Indeed, if u and v rewind several chunks off their transcript with each-other, then u must inform any other party $z \in \mathcal{N}(u)$ to rewind the same amount of chunks. This rewinding happens *even if the transcripts on the link* (u,z) *are consistent* at both ends.

Technically, if the transcript of u and v consists of k chunks, then u will send a "rewind" message to any neighbor z for which the transcript of u and z contains more than k chunks. However, there are a few caveats. First, any party z that is currently trying to find agreement with u via the meeting-points mechanism should not rewind the transcript with u. Intuitively, we can see that any such rewind seems unnecessary, since z is already going to truncate its transcript when it eventually finds agreement with u in the meeting-points subroutine. Furthermore, an underlying assumption of the meeting-points protocol is that, until the parties decide to truncate their transcripts in the protocol, their transcripts do not change. This is crucial since the MP mechanism may span over many rounds, and any abort (caused by a rewind) may make all this progress void.

Additionally, we restrict each party to rewinding at most one chunk in each of its pairwise transcripts. This is primarily for ease of analysis: it means that no matter what kind of errors the adversary induces, there is only so much harm that can be done during the rewind phase. The upshot of this is that it is not necessarily true that after the rewind phase, u sees exactly the same amount of simulated chunks with all $z \in \mathcal{N}(u)$.

Once a party u sends a rewind message to a neighbor z, party z will truncate one chunk of the transcript that corresponds to the link (u, z), and might then want to send rewind messages to its own neighbors. These rewinds could trigger more rewinds, leading to a wave of rewinds going through the network. By providing n rounds in the rewind phase, we make sure that this wave has enough time to go through the entire network.[8] This is critical to guaranteeing that we fix errors quickly enough to simulate Π with constant overhead.

The coding scheme

We now formally describe the coding scheme assuming a common random string (CRS) and oblivious noise (Algorithm 1). Let Π be a noiseless protocol over $G = (V, E)$, with R rounds and C transmissions throughout. Assume that the communication pattern and amount is predetermined, and independent of the parties' inputs and the transcript. Namely, let $T_\Pi = (m_1 m_2 \cdots m_C)$ be the noiseless transcript of Π; the content of the messages m_i depend on the specific inputs, however their order, source and destination are fixed for Π.

We partition T_Π into rounds according to Π, and group the rounds into *chunks*, where each chunk is a set of contiguous rounds with total communication complexity exactly $5K$. Specifically, we keep adding rounds to a chunk until adding a round would cause the communication to exceed $5K$. Note that without the last round, the communication in the chunk is at least $5K - 2m + 1$ bits. We can then add a virtual round that makes the communication in the chunk be exactly $5K$ bits. This addition affects the communication complexity by a constant factor. From this point on, we assume that Π adheres to our

[8] Alternatively, we could have fixed the rewind phase to consist of D rounds (rather than n rounds), where D is the the diameter of the graph G.

Algorithm 1 A noise-resilient simulation of Π (for party u)

```
Let $T_{u,v}$ denote the partial, pairwise transcript between $u$ and $v$ *according to $u$*, and let $|T_{u,v}|$
denote the number of *chunks* simulated so far in $T_{u,v}$.

1:  InitializeState( )

2:  for $i = 1$ to $100|\Pi|$ do
3:      for all $v \in \mathcal{N}(u)$ in parallel do                          ▷ meeting points
4:          $status_{u,v} \leftarrow$ MeetingPoints$(u, v, S_{i,u,v})$
5:      $minChunk \leftarrow \min_{v\in\mathcal{N}(u)} |T_{u,v}|$
6:      if exists $v$ such that $status_{u,v}$ = "meeting points" then
7:          $status_u \leftarrow 0$
8:      else if exists $v$ such that $|T_{u,v}| > minChunk$ then
9:          $status_u \leftarrow 0$
10:     else
11:         $status_u \leftarrow 1$

12:     $netCorrect_u \leftarrow$ FlagPassing$(u, status_u)$                    ▷ flag passing

13:     if $netCorrect_u = 1$ then                                            ▷ simulation
14:         Listen for one round.
15:         Simulate chunk $|T_{u,v}| + 1$ with each party $v \in \mathcal{N}(u)$ from whom we
            have not received $\perp$ in line 14. The simulation is based on the partial
            transcript $T_{u,w}$ for each $w \in \mathcal{N}(u)$, as well as the input to $u$.
16:         If the above step took less than $5K$ rounds, wait until $5K$ rounds have passed
17:         if received no $\perp$'s in Line 14 in this iteration then
18:             $minChunk \leftarrow minChunk + 1$
19:     else
20:         Send a single $\perp$ to each neighbor, and wait $5K$ rounds.

21:     for round $r = 1$ to $n$ do                                            ▷ rewind
22:         for all $v \in \mathcal{N}(u)$ in parallel do
23:             if $status_{u,v} \neq$ "meeting points" AND $alreadyRewound_{u,v} = 0$ then
24:                 if $|T_{u,v}| > minChunk$ then
25:                     Send a rewind message to $v$ and truncate $T_{u,v}$ by one chunk
26:                     $alreadyRewound_{u,v} \leftarrow 1$
27:             if a rewind message is received from $v$ then
28:                 if $status_{u,v} \neq$ "meeting points" AND $alreadyRewound_{u,v} = 0$ then
29:                     Truncate $T_{u,v}$ by one chunk
30:                     $alreadyRewound_{u,v} \leftarrow 1$
```

Algorithm 2 InitializeState()

```
1: $K \leftarrow m$
2: for all neighbors $v \in \mathcal{N}(u)$ do
3:     Initialize $T_{u,v} = \varnothing$
4:     Initialize $k_{u,v}, E_{u,v}, mpc1_{u,v}, mpc2_{u,v} \leftarrow 0$
5:     $status_{u,v} \leftarrow$ "simulate"
6:     $alreadyRewound_{u,v} \leftarrow 0$
7:     $S_{u,v} := (S_{i,u,v})_{i\in[100|\Pi|]} \overset{\text{unif}}{\leftarrow} \left(\{0,1\}^{\Theta(|\Pi|K)}\right)^{100|\Pi|}$ uniform bits of randomness.
8: $status_u \leftarrow 1$.
9: $netCorrect_u \leftarrow 1$
```

Algorithm 3 FlagPassing(u, $status$)

Let $\rho \in V$ be a specific node known by all the parties. Let $\mathcal{T}$ be a spanning tree generated by a breadth-first-search starting from ρ. Denote the depth of $\mathcal{T}$ as $d(\mathcal{T})$, where the depth of a single vertex is 1. Finally, let the *level* of a vertex be defined as $\ell(v) := \ell(u) + 1$, where u is the parent of v in $\mathcal{T}$, and $\ell(\rho) = 1$.

```
1: $netCorrect \leftarrow status$
2: if $u$ is a leaf in $\mathcal{T}$ then
3:     Send $netCorrect$ to parent vertex in $\mathcal{T}$.
4:     Sleep for $\ell(u) - 1$ rounds.
5: else
6:     Sleep for $d(\mathcal{T}) - \ell(u)$ rounds. Ignore any messages received in these rounds.
7:     Receive $b_1, \ldots, b_k$, one symbol from each child in $\mathcal{T}$.
8:     $netCorrect \leftarrow \bigwedge_{i=1}^{k} b_i \wedge status$
9:     if $u \neq \rho$ then
10:        Send $netCorrect$ to parent.
11:        Sleep for $\ell(u) - 1$ rounds. Ignore any messages received in these rounds.
12: if $u = \rho$ then
13:     Send $netCorrect$ to children.
14: else
15:     Sleep for $\ell(u) - 1$ rounds. Ignore any messages received in these rounds.
16:     Receive $b$ from parent.
17:     $netCorrect \leftarrow b \wedge status$
18:     if $u$ is not a leaf in $\mathcal{T}$ then
19:         Send $netCorrect$ to children.
20:         Sleep for $d(\mathcal{T}) - \ell(u)$ rounds. Ignore any messages received in these rounds.
21: return $netCorrect$
```

required structure.

We number the chunks in order, starting from 1. For any (possibly partial) transcript T, we let $|T|$ denote the *number of chunks* contained in the transcript T. In particular, $|\Pi|$ is the maximal number of chunks in Π. We assume without loss of generality that in each chunk, each party sends at least one bit to each of its neighbors (again, this can easily be achieved by pre-processing Π while increasing its communication by a constant factor). In addition, we assume that the protocol Π is padded with enough dummy chunks where parties simply send zeros. Concretely, we assume the number of chunks is $100|\Pi|$, where the last $99|\Pi|$ chunks are all padding. This padding is standard in the literature on interactive coding, and is added to deal with the case that the adversary behaves honestly in all the rounds until the last few rounds, and fully corrupts the last few rounds.

The parties simulate Π one chunk at a time, by cycling through the following phases in the following order: consistency check, flag passing, simulation, and rewind. Each phase takes a number of rounds that is a priori fixed, and since our model is synchronous, the parties are always in agreement regarding which phase is being executed.

Let $T_{u,v}$ denote the pairwise transcript of the link (u,v) as seen by u, where $v \in \mathcal{N}(u)$; similarly, $T_{v,u}$ is the transcript of the same link as seen by v (which may differ from $T_{u,v}$ due to channel noise). In more detail, $T_{u,v}$ is the concatenation of the transcripts generated at each chunk, where the transcript of chunk i consists of two parts: (1) the simulated communication of chunk i, and (2) the chunk number i.[9] The structure of part (1) is as follows. Assume that in the i-th chunk in Π, j bits are exchanged over the (u,v) link in rounds $t_1,\dots,t_j$. Then $T_{u,v}$ holds a string of length j over $\{0,1,*\}$ describing the communication at times $t_1,\dots,t_j$, *as observed by* u. The symbol $*$ denotes the event of not receiving a bit at the specific round (i.e., due to a deletion). The transcript $T_{v,u}$ is defined analogously from v's point of view. Note that restricted to the substrings that belongs to chunk i, $T_{u,v} = T_{v,u}$ if and only if there where no errors at rounds $t_1,\dots,t_j$ in the simulation phase; insertions and deletions at other rounds are ignored. We abuse notation and define

[9] It is important to add the chunk number since the inner-product hash function we use (Definition 2.1.2) has the property that for any string x, $h(x) = h(x \circ 0)$.

$|T_{u,v}|$ to be the number of *chunks* that appear in $T_{u,v}$.

In Algorithm 1 we describe the noise-resilient protocol for a fixed party u. The parties start by initializing their state with a call to InitializeState() (Algorithm 2). The variables in line 4 of InitializeState() are used for keeping state across iterations of the meeting-points mechanism, described in Algorithm 11 (in Section 2.4).

Next, the parties perform a single iteration of the meeting-points mechanism (Algorithm 11). Given a pair of adjacent parties u and v, the meeting-points mechanism outputs a variable $status_{u,v}$, which indicates whether the parties want to simulate (in which case $status_{u,v}$ = "simulate") or continue with the meeting-points mechanism (in which case $status_{u,v}$ = "meeting points").

Then, according to the output of the meeting-point mechanism and according to any apparent inconsistencies in the simulated transcripts with its neighbors, each party sets its "flag" $status_u$ to denote whether it should continue with the simulation or not. This status is used as an input to the flag-passing phase, described in Algorithm 3, where $status_u = 0$ denotes that a "stop" flag should be propagated.

Each party ends the flag-passing phase with a flag denoted $netCorrect_u$ that is set to 1 if the network as a whole seems to be correct. Then, the parties perform a simulation phase. If the $netCorrect$ flag is set to 1, they execute Π for one additional chunk, according to the place they believe they are at. Otherwise, they send a special symbol $\perp$ to indicate they are not participating in the current simulation phase.

Finally, the rewind phase begins, where any party u that sees an obvious discrepancy in the lengths of the transcripts in its neighborhood, sends a single rewind request to any neighbor v which is ahead of the rest, conditioned that u and v are not currently in the middle of a meeting-points process.

2.2.4 Analysis via a Potential Function

In this section we analyze the coding scheme presented in Section 2.2.3 and lay the groundwork for proving the following theorem.

Theorem 2.2.2. *Let $G = (V, E)$ be a network with $n = |V|$ parties and $m = |E|$ links, and let Π be a multiparty protocol on the network G with communication complexity $\mathsf{CC}(\Pi)$, binary alphabet and fixed order of speaking. Let $|\Pi| := \frac{\mathsf{CC}(\Pi)}{5m}$ and let $\varepsilon > 0$ be a sufficiently small constant. Then, with probability at least $1 - \exp(-\Omega(|\Pi|))$, Algorithm 1 simulates Π correctly with communication complexity $\Theta(\mathsf{CC}(\Pi))$, assuming an oblivious adversary (fixing or additive) with error rate ε/m.*

We emphasize that Theorem 2.2.2 applies to both fixing and additive oblivious adversaries (see Section 2.1.2 for definitions). In fact, all we need here is for the transcripts to be independent of the seeds used to hash them; see Section 2.2.5 for an expanded discussion.

In order to prove the above theorem we define a *potential function* that measures the progress of the simulation at every iteration. Below, we define the potential function and intuitively explain most of its terms. Then, we prove that in every iteration[10] the potential increases by at least K, while the communication increases by *at most* $K \times \ell$, where ℓ measures the number of channel noise and hash collisions that occurred in that specific iteration.

We split the analysis of the potential into two parts: the meeting points mechanism and the rest of the coding scheme. The first part, regarding the meeting points mechanism, reiterates the analysis of [70] with minor adaptations. We defer the full proofs to Section 2.4. The rest of the potential analysis is novel. First, we focus on the iterations with no errors/hash-collisions, and show that the potential rises in this special case. Then, we focus on iterations that suffer from errors/hash-collisions. Then, in Section 2.2.5, we bound (with high probability) the number of hash-collisions that may happen throughout the entire execution of the coding scheme. Finally, in Section 2.2.6 we complete the proof of Theorem 2.2.2, using the fact that the potential at the end of the coding scheme must be high enough to imply a correct simulation of Π, given the bounded amount of errors and hash-collisions.

In the following, all our quantities measure progress in chunks, where each chunk

[10]Recall that a single iteration of Algorithm 1 consists of a consistency phase, flag-passing phase, simulation phase and a rewind phase.

contains exactly $5K = 5m$ bits. Recall that we denote by $|\Pi|$ the number of chunks in the noiseless protocol Π, and we denote by $|T_{u,v}|$ the number of *chunks* in the simulated (partial) transcript $T_{u,v}$.

The potential function

Our potential function ϕ will measure the progress of the network towards simulating the underlying interactive protocol Π correctly. Naturally, ϕ changes as the simulation of Algorithm 1 progresses, and so depends on the round number. In what follows, for ease of notation, we omit the current round number in all the terms used to define ϕ.

For each adjacent pair of parties u and v, define

$$G_{u,v} \tag{2.1}$$

to be the size (in chunks) of the longest common prefix of $T_{u,v}$ and $T_{v,u}$. Namely, $G_{u,v}$ is the length of the largest prefix of communication between parties u and v in Π, that these parties agree on. Define $B_{u,v}$ to be

$$B_{u,v} \overset{\text{def}}{=} \max(|T_{u,v}|, |T_{v,u}|) - G_{u,v}. \tag{2.2}$$

Namely, $B_{u,v}$ is the gap between how far one of the parties *thinks* they have simulated and how far they have simulated correctly.[11] Note that $B_{u,v}$ is always nonnegative by design. Furthermore, $B_{u,v} = 0$ if and only if the parties have no differences in their pairwise transcripts with each other.

Define

$$G^* \overset{\text{def}}{=} \min_{(u,v)\in E} G_{u,v} \tag{2.3}$$

to be the largest chunk number through which the network *as a whole* has correctly

[11] Note that we can have $B_{u,v} = 0$ even when there have been errors in the network, as long as those errors were corrected.

simulated. Let

$$H^* \overset{\text{def}}{=} \max_u \max_{v \in \mathcal{N}(u)} |T_{u,v}| \tag{2.4}$$

denote the largest chunk number which any party in the network *thinks* it has simulated; note that, by definition, $H^* \geq G^*$. Finally, we define

$$B^* \overset{\text{def}}{=} H^* - G^*. \tag{2.5}$$

In addition, our potential function also quantifies the progress of the meeting-points mechanism between any two adjacent parties in the network (which we elaborate on below, and in Section 2.4). This is done via the term $\varphi_{u,v}$ defined in Eq. (2.42) in Section 2.4.3, which is closely inspired by the potential function stated in [70]. Intuitively, $\varphi_{u,v}$ is the number of iterations of the meeting-points mechanism that parties u and v need to do to make $B_{u,v} = 0$; indeed, for all pairs $(u,v) \in E$ it holds that (Proposition 2.4.2)

$$0 \leq B_{u,v} \leq \varphi_{u,v},$$

and in particular, $\varphi_{u,v} = 0$ implies that $B_{u,v} = 0$.

Finally, let EHC denote the number of errors and hash collisions that have occurred in the protocol until the current round of Algorithm 1. Similarly to all the other terms in the potential, we drop the dependence on the round r.

Our potential function is defined to be:

$$\phi \overset{\text{def}}{=} \sum_{(u,v)\in E} \left(\frac{K}{m} G_{u,v} - K \cdot \varphi_{u,v} \right) - C_1 K B^* + C_7 K \cdot \mathsf{EHC} \tag{2.6}$$

where C_1 and C_7 are constants such that C_1 is sufficiently larger than 2, but smaller than all the constants $C_2, \ldots, C_6$ defined in Eq. (2.42), and C_7 is a constant sufficiently larger than $C_2, \ldots, C_6$. We refer the reader to Figure 2-1 for a summary of the definitions of all variables defined in this section.

Parameter	**Definition**
$T_{u,v}$	Transcript of communication between u and v according to u
$G_{u,v}$	Size of longest common prefix of $T_{u,v}$ and $T_{v,u}$ (in chunks)
$B_{u,v}$	$\max(\lvert T_{u,v}\rvert, \lvert T_{v,u}\rvert) - G_{u,v}$
G^*	$\min_{(u,v)\in E} G_{u,v}$
H^*	$\max_u \max_{v\in\mathcal{N}(u)} \lvert T_{u,v}\rvert$
B^*	$H^* - G^*$
$\varphi_{u,v}$	Meeting points potential between u and v
EHC	Number of errors and hash collisions that have occurred overall
ϕ	Overall potential in network

Figure 2-1: *Definition of terms related to the potential function ϕ.*

The meeting-points mechanism and potential $\varphi_{u,v}$

In what follows, we briefly recall the meeting-points mechanism and why we use it. We defer the formal definition of $\varphi_{u,v}$ and all the proofs regarding it to Section 2.4.

If two adjacent parties u and v have $T_{u,v} \neq T_{v,u}$ (or equivalently, $B_{u,v} > 0$), then they should not simulate further with each other without fixing the differences in their transcripts. If u and v knew which of them needs to roll back and by how much, they could simply roll back the simulated chunks until $T_{u,v} = T_{v,u}$, at which point they can continue the simulation. However, they do not know this information. Furthermore, they cannot afford to communicate $|T_{u,v}|$ or $|T_{v,u}|$, since these numbers potentially require $\log|\Pi|$ bits to communicate.

This problem is solved via the "meeting-points" mechanism [70] which is designed to roll back $T_{u,v}$ and $T_{v,u}$ to a point where $T_{u,v} = T_{v,u}$, while only requiring $O(B_{u,v})$ exchanges of hashes between parties u and v, and guaranteeing that (in the absence of error) neither u nor v truncate their transcript too much. That is, u (resp. v) truncates $T_{u,v}$ (resp. $T_{v,u}$) by at most $2B_{u,v}$ chunks. While errors and hash collisions can mess up this guarantee, each error or hash collision causes only a bounded amount of damage. Since the adversary's allowed error rate is sufficiently small, the simulation overcomes this damage with high probability.

As mentioned above, our analysis of the meeting-points mechanism essentially follows that of Haeupler [70] after adopting it to our construction, where the meeting-points mechanism is interleaved over several iterations, rather than performed all at once. Specifically, for each link $(u,v) \in E$ we define a "meeting-points potential" term $\varphi_{u,v}$ that approximately measures the number of hash exchanges it will require for u and v to roll back $T_{u,v}$ and $T_{v,u}$ to a common point. While our analysis of how $\varphi_{u,v}$ changes in the meeting-points phase naturally repeats the analysis of [70], $\varphi_{u,v}$ can also change during the other phases of the protocol, especially when noise is present. Our analysis bounds the change in $\varphi_{u,v}$ in all the phases as a function of the errors and hash collisions that occur throughout the iteration. This allows us to bound the change in the overall potential ϕ. We bound the changes in $\varphi_{u,v}$ in the Flag Passing, Rewind, and Simulation phases in Claim 2.4.1. The changes in $\varphi_{u,v}$ in the Meeting Points phase are addressed in Lemma 2.4.6 (analogous to Lemma 7.4 in [70]) and Proposition 2.4.4, and are combined to establish how the potential ϕ changes in the Meeting Points phase (Lemma 2.4.11).

We defer the formal definition of the meeting-points mechanism and the proofs of the relevant properties to Section 2.4.

Bounding the potential increase and communication per iteration

In this section we prove the following technical lemma that says that the potential ϕ (Eq. (2.6)) increases in each iteration by at least K. Furthermore, the amount of communication performed during a single iteration can be bounded by roughly K times the amount of links (i.e., pairs of parties) that suffer from channel-noise during this iteration, or links that experienced an event of hash-collision during this iteration.

Lemma 2.2.3. *Fix any iteration of Algorithm 1 and let ℓ be the number of links with errors or hash collisions on them during this iteration. Then,*

1. *The potential ϕ increases by at least K in this iteration.*
2. *The amount of communication in the entire network during this iteration* (CC) *satisfies*

$$\mathsf{CC} \le \alpha(1+\ell)K,$$

where α is a sufficiently large constant.

We turn to proving the above lemma. Let us begin by giving a high-level overview.

Proof Overview We proceed to prove the lemma in two conceptual steps.

1. First, we consider iterations that have no errors or hash collisions.

   We first establish that the communication in this case is at most $O(K)$. To this end, we first argue that the communication in the meeting-points, flag-passing, and rewind phases is *always* bounded by $O(K)$ (Proposition 2.2.4), regardless of errors committed by the adversary. Therefore, it suffices to bound the communication in the simulation phase. If every party is simulating the same chunk, then the communication is easily bounded by $O(K)$. However, if the parties are simulating many different chunks, then the communication could be much larger. This is where the flag-passing phase is useful: if there are no errors, then the flags will prevent all parties from simulating when two parties are at different chunks.

   We next establish that the potential increases by at least K, as follows. If the parties simulate, then since there are no errors or hash collisions, $\sum G_{u,v}$ increase by K, and none of the other terms change. If the parties do not simulate, then either some adjacent parties did not pass their consistency check, in which case $\varphi_{u,v}$ increases by $\Omega(1)$ (Lemma 2.4.6) and none of the other terms change, or some parties rewind, in which case B^* decreases and none of the other terms change.

2. Next,we consider iterations that have errors or hash collisions.

   We first argue that errors and hash collisions increase ϕ by at least K. To this end, note that errors may cause some terms of ϕ to decrease, but this is compensated for by the accompanying increase in EHC, and since C_7 is set to be large enough, even though some of the terms decrease, overall the potential increases by at least K.

   We would then like to argue that the communication increases by at most $O(K)$, though unfortunately, this claim is false. The communication in an iteration can actually greatly exceed $O(K)$, though we show that in these cases, there were many errors or hash collisions in the iteration. Specifically, we argue that each error or hash collision *individually* does not cause too much extra communication. This is formalized in Lemma 2.2.9.

Iterations with no errors or hash collisions First, we prove a simple proposition, which says that the communication in the meeting-points, flag-passing, and rewind phases is bounded. This reduces bounding the overall communication in an iteration to bounding the communication in the corresponding simulation phase.

Proposition 2.2.4. *The communication during the flag-passing and rewind phases is $O(m)$ in total, and the communication in the meeting-points phase is $O(K)$, regardless of errors or hash collisions in the iteration.*

Proof. In the meeting-points phase, each adjacent pair of parties exchange hashes of their transcripts (see Algorithm 11), where the output length of the hash functions is $\Theta(K/m)$. Hence, there is $O(K)$ communication in the meeting-points phase.
The communication pattern in the flag-passing phase is deterministic and consists of two messages per link of a the spanning tree $\mathcal{T}$, hence it is upper bounded by $O(n) = O(m)$. Finally, each link can have at most one valid "rewind" message in the rewind phase (note that messages that are inserted do not count towards our communication bound). □

Lemma 2.2.5. *Suppose that there are no errors or hash collisions in a single iteration of Algorithm 1. Then the overall communication in the network is $O(K)$.*

Proof. By Proposition 2.2.4, the communication in all the phases except of the simulation phase are bounded by $O(K)$, and we are left to bound the communication in the simulation phases

In the simulation phase, each party either sends $\perp$ or simulates a specific chunk. Say that v simulates chunk number n_v with all its neighbors if it didn't send $\perp$ (however, its neighbors may simulate a different chunk). Each chunk contains at most $5K$ bits of communication, hence, the total amount of communication in the simulation phase is bounded by $5K$ times the number of distinct chunk numbers being simulated in the network. In other words, it is bounded by $5K \cdot |\{n_v \mid v \in V\}|$, up to additional $2m$ $\perp$ "messages" (which in our case are merely $2m$ bits).

Therefore, to finish the proof it remains to argue that if there are no errors or hash collisions then $|\{n_v \mid v \in V\}| = 1$ and the potential function increases by at least K in the iteration. We consider several different cases according to the state of the network at the beginning of the iteration, specifically, whether the parties have set $netCorrect = 1$ or not.

Case 1: At the end of the flag-passing phase, $netCorrect_u = 1$ **for every party** u**.** Since there were no errors or hash collisions, the fact that $netCorrect_u = 1$ means that each party u had $status_u = 1$ before the flag-passing phase. This follows since by the definition of the flag-passing phase, for every party $v \in V$, $netCorrect_v = \bigwedge_{u \in V} status_u$. Note that, since we assume that none of the parties have $netCorrect_u = 0$, and we assume no errors, the are no $\perp$ symbols sent in the simulation phase.

The fact that each party u has $status_u = 1$ implies that for all $v, w \in \mathcal{N}(u)$, it holds that $|T_{u,v}| = |T_{u,w}|$. Further, for any $v \in \mathcal{N}(u)$, $T_{u,v} = T_{v,u}$, or otherwise the hashes would indicate a mismatch and the parties would have set $status = 0$. Putting these two facts together, we get that $G^* = H^*$ and hence $B^* = 0$, which implies that indeed $|\{n_v \mid v \in V\}| = 1$, as desired.

Case 2: At the end of the flag-passing phase, some party u **has** $netCorrect_u = 0$**.** Since $netCorrect_u = 0$ for some party u, there must be some party v such that $status_v = 0$, and hence we have that $netCorrect_u = 0$ for all $u \in V$. Since $netCorrect_u = 0$ for all parties u, we know that none of the parties will simulate (they will only send $\perp$s) and hence the overall communication in the iteration will be $2m = O(K)$.

□

We next show that the potential increases by at least K in any such iteration.

Lemma 2.2.6. *Suppose that there are no errors or hash collisions in a single iteration of Algorithm 1. Then the potential ϕ increases by at least K during this iteration.*

Proof. We consider the status of the network at the iteration according to the next three cases.

Case 1: At the end of the flag-passing phase, $netCorrect_u = 1$ **for every party** u**.** Recall that since there were no errors or hash collisions, the fact that $netCorrect_u = 1$ means that each party u had $status_u = 1$ before the flag-passing phase. This in turn implies that for all $v, w \in \mathcal{N}(u)$, it holds that $|T_{u,v}| = |T_{u,w}|$. Further, for any $v \in \mathcal{N}(u)$, $T_{u,v} = T_{v,u}$, or otherwise the hashes would indicate a mismatch and the parties would have set $status = 0$. Consequently, we have $G^* = H^*$ and $B^* = 0$. The fact that $netCorrect_u = 1$ for every party u, together with the fact that $B^* = 0$, implies that all parties simulate the same chunk, and the absence of errors in the communication implies that this simulation is done correctly. Hence, each $T_{u,v}$ is extended correctly according to Π. This in turn implies that $G_{u,v}$ increases for each $(u,v) \in E$, which causes ϕ to increase by K.

Next, we argue that none of the other terms of ϕ decrease. We first argue that B^* remains zero at the end of the iteration. To this end, note that since all parties simulate one chunk in each of their pairwise transcripts, we still have the property that $|T_{u_1,v_1}| = |T_{u_2,v_2}|$ for all $(u_1, v_1) \in E$ and $(u_2, v_2) \in E$ after the simulation phase. Since there were no errors, we also have that $T_{u,v} = T_{v,u}$ for all $(u,v) \in E$. As noted before, this gives us that $B^* = 0$ after the simulation phase, and since there are no errors it remains zero after the rewind phase as well.

It remains to argue that $\varphi_{u,v}$ does not increase for any $(u,v) \in E$. By Proposition 2.4.4 we know that $\varphi_{u,v}$ does not increase in the meeting-points phase. Furthermore, it does not increase in the flag-passing, simulation or rewind phases either, by Claim 2.4.1.

Putting this all together, we have that each $G_{u,v}$ increases by one, B^* does not change and $\varphi_{u,v}$ does not increase, which implies that the potential ϕ increases by at least K overall, as desired.

Case 2: Some party u has a neighbor $v \in \mathcal{N}(u)$ s.t. $status_{u,v}$ = "meeting points" after the meeting-points phase. Since $status_{u,v}$ = "meeting points", u has $status_u = 0$ after the meeting-points phase, and therefore, given the lack of errors, each party $x \in V$, will set $netCorrect_x = 0$ after the flag-passing phase. Note that since none of the parties are simulating the next chunk during the simulation phase, it follows that ϕ does not change in the simulation phase. Next note that the potential increases by at least $5K$ during the meeting-points phase (Lemma 2.4.11). Since the potential does not change during the flag-passing phase, it remains to argue that the potential function does not decrease by much during the rewind phase.

In the rewind phase, we may have parties that send rewinds. Even though these rewinds seem to take us in the right direction, they may cause a small decrease in some terms of the potential. However, we argue that in the rewind phase the potential decreases by at most K, and thus in total, ϕ increases by at least $5K - K = 4K$, as desired.

To this end, first note that since we limit the number of truncations per link to at most one, it follows that $(K/m)\sum_{(x,y)\in E} G_{x,y}$ can decrease by at most K. It remains to argue that B^* and $\{\varphi_{x,y}\}_{(x,y)\in E}$ do not increase in the rewind phase.

The fact that $\varphi_{x,y}$ does not increase follows from Claim 2.4.1. To argue that B^* does not increase, note that since no party simulates in this iteration, H^* does not increase. We claim a party x will never truncate a transcript $T_{x,y}$ such that $|T_{x,y}| = G^*$. Clearly x will not send a rewind message to y, since there is no y^* such that $|T_{x,y^*}| < |T_{x,y}|$ by the definition of G^*. We also claim that y will not send a rewind message to x. If $|T_{y,x}| = |T_{x,y}| = G^*$, then this follows because there is no x^* such that $|T_{y,x^*}| < |T_{y,x}|$. Otherwise if $|T_{y,x}| \neq |T_{x,y}|$, then since there were no hash collisions or errors in the meeting-points phase we conclude that $status_{y,x}$ = "meeting points". Therefore y will not send a rewind message to x.

Case 3: At end of the meeting-points phase, $status_{u,v}$ = "simulate" for all $(u,v) \in E$, yet at the end of the flag-passing phase, some party u has $netCorrect_u = 0$. Again, since $netCorrect_u = 0$ for some party u, there must be some party v such that $status_v = 0$, and hence we have that $netCorrect_u = 0$ for all $u \in V$. No party simulates on the next simulation phase (they will only send $\perp$s), hence, the potential does not change during the simulation phase. In addition, the potential does not decrease in the meeting-points phase (Lemma 2.4.11). Furthermore, the potential remains unchanged during the flag-passing phase. Therefore, all that remains to show is that ϕ increases in the rewind phase by at least K.

Recall that by Claim 2.4.1, $\varphi_{u,v}$ will not increase during the rewind phase for any (u,v). Furthermore, $\sum G_{u,v}$ can decrease by at most m in the rewind phase, which means $(K/m)\sum G_{u,v}$ decreases by at most K. Therefore, it suffices to show that B^* decreases by 1, and therefore that ϕ increases by $C_1K - K \geq K$, since $C_1 \geq 2$.

To this end, we first show that G^* does not decrease, and then show that H^* decreases by 1. For the former, note that a party u will never issue a rewind to a party v for which $|T_{u,v}| = G^*$, since this would mean that there is some party $w \in \mathcal{N}(u)$ such that $|T_{u,w}| < G^*$, which contradicts the definition of G^*. Therefore, G^* does not decrease.

To argue that H^* decreases by 1, fix a party u and a neighbor v such that $|T_{u,v}| = H^*$. We argue that during the rewind phase party u will rewind this transcript by one chunk. The proof goes by induction on the distance between u and a party whose $minChunk \neq H^*$. To this end, let $S^* = \{v : \exists w \in \mathcal{N}(v) : |T_{v,w}| < H^*\}$ be the set of parties that have some transcript below chunk H^*.

Claim 2.2.7. *S^* is non-empty.*

Proof. Indeed, given that all parties have $netCorrect_v = 0$ after the flag-passing phase, despite all pairs having $status_{u,v}$ = "simulate", it must hold that some party sees an inconsistency in the lengths of the simulated transcript with two of its neighbors (Line 8). Namely, for some party p, there are neighbours w, w' such that $|T_{p,w}| \neq |T_{p,w'}|$. It follows that $|T_{v,w}| = H^*$ cannot hold for all $(v,w) \in E$. □

Let $d(u, S^*)$ denote the shortest distance in the graph G between a party u and some party in S^*, where $d(u, S^*) = 0$ if and only if $u \in S^*$.

Claim 2.2.8. *Party u truncates $T_{u,v}$ after at most $d(u, S^*) + 1$ rounds of the rewind phase.*

Proof. Note that if $d(u, S^*) = 0$, i.e., $u \in S^*$, then in the next round of the rewind phase for every neighbor $w \in \mathcal{N}(u)$ we have $|T_{u,w}| < H^*$. This is the case since u has $minChunk_u < H^*$ by the definition of S^*. Then, u sends a rewind message to any w with $|T_{u,w}| = H^*$, and truncates that transcript (Line 24).

Next we claim that if $d(u, S^*) = j$ for some $j > 0$ at the beginning of some round r in the rewind phase, then in the beginning of round $r+1$ it holds that $d(u, S^*) = j-1$. Let $u, a_1, a_2, \dots, a_j \in S^*$ be the vertices in a shortest path from u to S^*. Note that for any two consecutive parties along this path, $|T_{a_i,a_{i-1}}| = |T_{a_{i-1},a_i}| = H^*$, and the same holds for u and a_1. This is true since $u, a_1, \dots, a_{j-1} \notin S*$, and since and $status_{a_j,a_{j-1}} = status_{a_{j-1},a_j} =$ "simulate", which means any two parties are consistent with their transcripts.

Since $a_j \in S^*$ we have that $minChunk_{a_j} < H^*$ and it follows that in round r, party a_j sends a rewind message to a_{j-1} (Line 24). We stress that no rewind message has yet been sent on the link (a_j, a_{j-1}). Indeed, if this were not the case, then we would have gotten $|T_{a_j,a_{j-1}}| < H^*$ already in a prior round where the rewind message took place. But this contradicts $a_{j-1} \notin S^*$ in the beginning of round r. Hence, by the end of round r we have that $|T_{a_j,a_{j-1}}| = |T_{a_{j-1},a_j}| = H^* - 1$ and thus $a_{j-1} \in S^*$. This means that $d(u, S^*) = j-1$ at the beginning of round $r+1$.

By employing the same argument inductively we get that, if at the beginning of the rewind phase $d(u, S^*) = j$, then after j rounds we have that $u \in S^*$, and after the $(j+1)-th$ round, party u has truncated $T_{u,v}$ by at least one chunk, as needed. □

Showing that H^* has decreased by 1 is now straightforward. For any party u we note that $d(u, S^*)$ can never exceed $n-1$, which is an upper bound of the diameter of G. Since the rewind phase consists of n rounds, after its $(n-1)$-th round, all parties are in S^*, and by the end of the n-round of the rewind phase, all pairwise transcripts are of length at most $H^* - 1$. This completes the proof. □

Iterations with errors and hash collisions

Lemma 2.2.9. *Let ℓ be the number of links that experienced either errors or hash collisions* *during a given iteration, and assume $\ell \geq 1$. Then the increase in the potential ϕ in this iteration is at least $\Omega(C_7 \ell K)$, and the amount of communication in this iteration is at most $O(\ell K)$.*

Proof. Let ℓ_1 denote the number of links with errors and hash collisions in the meeting-points phase, let ℓ_2 denote the number of links with errors in the flag-passing phase, let ℓ_3 denote the number of links with errors in the simulation phase, and let ℓ_4 denote the number of links with errors in the rewind phase. Then $\ell \leq \ell_1 + \ell_2 + \ell_3 + \ell_4$.

Let us begin with bounding the communication in this iteration. By Proposition 2.2.4, the communication in all the phases except for the simulation phase are bounded by $O(K)$, and we are left to bound the communication in the simulation phase. As explained in the proof of Lemma 2.2.5, the amount of communication in the simulation phase is bounded by $5K$ times the number of distinct chunk numbers being simulated in the network (plus $2m$ for $\perp$s). We show that this number is proportional to the number links that experienced errors or hash collisions during that same iteration.

Let $\mathcal{T}$ be the spanning tree used for the flag passing in Algorithm 3. Consider the subgraph $\mathcal{H}$ of $\mathcal{T}$ induced by only keeping an edge $(u,v) \in \mathcal{T}$ if $netCorrect_u = netCorrect_v = 1$ and $|T_{u,v}| = |T_{v,u}|$. Recall that $netCorrect_u = 1$ implies that $status_u = 1$, and thus for any u with $netCorrect_u = 1$ we know that $|T_{u,v}| = |T_{u,w}|$ for all $w \in \mathcal{N}(u)$ and in particular for all w such that $(u,w) \in \mathcal{H}$. By a straightforward induction, one can argue that for any pair of parties u and x that are in the same connected component of $\mathcal{H}$, it holds that $|T_{u,v}| = |T_{x,y}|$ for any v, y s.t. $(u,v) \in \mathcal{H}$ and $(x,y) \in \mathcal{H}$. Hence, in any single connected component of $\mathcal{H}$, at most one chunk of Π is being simulated. Note that there are components in $\mathcal{H}$ with *no* chunk being simulated. Each such connected component consists of a single isolated variable u such that $netCorrect_u = 0$.

Claim 2.2.10. *Let S denote the set of connected components in $\mathcal{H}$ such that $netCorrect_u = 1$, and let $s = |S|$ be the number of components in S. Then,*

$$s - 1 \leq \ell_1 + \ell_2.$$

Proof. We claim that there are at least $s-1$ edges (u,v) in $\mathcal{T}\setminus\mathcal{H}$ such that v is in a component in S and $\ell(u) < \ell(v)$, where recall that $\ell(u)$ is defined to be the distance of u from ρ, which is the root of $\mathcal{T}$, plus 1. Towards seeing this, note that each connected component in S is a subtree of $\mathcal{T}$, and since they are disjoint, at least $s-1$ many of them do not have ρ as the root of the subtree. Let v be such a root, and let u be its parent in $\mathcal{T}$. This satisfies the desired conditions.

Fix such an edge (u,v). We argue that there was an error on the link (u,v) in either the meeting-points or flag-passing phases, establishing the claim. Since (u,v) is an edge in $\mathcal{T}\setminus\mathcal{H}$, we know that either $netCorrect_u = 0$, $netCorrect_v = 0$, or $|T_{u,v}| \neq |T_{v,u}|$; otherwise, (u,v) would have been in $\mathcal{H}$. However, we know that $netCorrect_v = 1$, since v is in a connected component in S and by the definition of S. Hence, it can either be that $netCorrect_u = 0$ or that $|T_{u,v}| \neq |T_{v,u}|$.

If $|T_{u,v}| \neq |T_{v,u}|$, then there must have been an error or hash collision in the meeting-points phase, otherwise we would have $status_v = 0$ implying $netCorrect_v = 0$, which is a contradiction. If, on the other hand, $netCorrect_u = 0$ holds, then there must have been an error in the downward part of the flag-passing phase, since $netCorrect_v = 1$, so clearly v did not correctly receive the flag that u sent. □

As argued above, the communication during the simulation phase is bounded by $s \cdot 5K + 2m$: each connected component in S jointly simulates a single chunk, and components outside of S (which consists of a single party u such that $netCorrect_u = 0$) do not simulate; the $2m$ term comes from potential $\perp$ messages. The above claim implies that $s \leq \ell_1 + \ell_2 = O(\ell)$, leading to communication of $O(\ell K)$ in the simulation phase. Since all the other phases have communication $O(K)$, the communication in the entire iteration is a as claimed.

To finish the proof of the Lemma it remains to bound the increase in the potential ϕ. Consider the various phases in the iteration, and the terms of ϕ given by Eq. (2.6).

- **Meeting Points:** Lemma 2.4.11 guarantees that the potential ϕ goes up by at least $5cK + 0.4C_7\ell_1 K$, where c is the number of pairs of parties (u,v) such that $(u,v) \in E$ and $status_{u,v}$ or $status_{v,u}$ is “meeting points” at the end of the Meeting Points phase.

- **Flag Passing:** No direct potential change happens in this phase, other than any increase in potential caused by an error induced by the adversary. So the potential in this phase increases by at least $C_7\ell_2 K$.

- **Simulation:** The term $\sum G_{u,v}$ cannot decrease in the simulation phase, since no transcript is being truncated in this phase. Claim 2.4.1 establishes that $\varphi_{u,v}$ increases by C_3 in the simulation phase only if there was an error on the link (u,v) somewhere in this iteration. Otherwise, it does not increase. Hence, the term $-\sum K \cdot \varphi_{u,v}$ decreases by at most $KC_3\ell$. The third term,$-C_1KB^*$, decreases by at most KC_1. Indeed, $B^* = H^* - G^*$ increases by at most 1 in the simulation phase, since H^* can increase by 1 but G^* cannot decrease, since no party truncates in this phase. The fourth term, $C_7K \cdot \mathsf{EHC}$, increases by at least $C_7\ell_3 K$. Thus in total, the potential function increases during the simulation phase by at least $-C_3\ell K - C_1K + C_7\ell_3 K$.

- **Rewind:** The term $(K/m)\sum G_{u,v}$ can decrease in the rewind phase by at most K, since each party rewinds a transcript at most one chunk. The second term $-\sum K \cdot \varphi_{u,v}$ decreases by at most $KC_3\ell$, again, by Claim 2.4.1. The third term $-C_1KB^*$ decreases by at most C_1K since B^* increases by at most 1 in the rewind phase: G^* can decrease by at most one (since no party rewinds more than one chunk) and H^* cannot increase. The fourth term $C_7K\mathsf{EHC}$ increases by $C_7\ell_4 K$.

Putting it all together, we get that in the entire iteration ϕ increases by at least

$$\begin{aligned}
&5cK + 0.4C_7\ell_1 K + C_7\ell_2 K - C_3\ell K - C_1K + C_7\ell_3 K - K - C_3\ell K - C_1K + C_7\ell_4 K \\
&\geq 5cK + 0.4C_7(\ell_1 + \ell_2 + \ell_3 + \ell_4)K - 2C_3\ell K - 2C_1K - K \\
&\geq 5cK + (0.4C_7 - 2C_3)\ell K - (2C_1 + 1)K
\end{aligned}$$

where the final inequality follows from the fact that $\ell_1 + \ell_2 + \ell_3 + \ell_4 \geq \ell$. Since $\ell \geq 1$ (by our assumption), we can take C_7 to be sufficiently large compared to C_3 and C_1, and get that the change in ϕ is $\Omega(C_7\ell K)$, as desired. $\square$

We can finally complete the proof of Lemma 2.2.3.

Proof of Lemma 2.2.3. We first recall that Lemma 2.2.6 establishes that, in the absence of errors and hash collisions, the potential increases by K and Lemma 2.2.5 bounds the total communication by $O(K)$. Now assume that there is at least one error or hash collision in the iteration, so $\ell \geq 1$. Then Lemma 2.2.9 shows that the potential increases by $\Omega(C_7 \ell K)$, and that the communication in the iteration is at most $O(\ell K)$. By taking C_7 to be sufficiently large, we can see that ϕ increases by at least K while the communication is bounded by $O((\ell+1)K)$, as required. $\square$

2.2.5 Bounding hash collisions and communication

In this section we prove that the number of hash collisions throughout the entire simulation is bounded by $O(\varepsilon|\Pi|)$ with high probability, where $|\Pi|$ is the number of chunks in the original, noiseless protocol.

The main lemma we prove in this part of the chapter is the following.

Lemma 2.2.11. *Let $\varepsilon > 0$ be a sufficiently small constant. Suppose a hash function h in Algorithm 1 with hash collision p where $p < \frac{1}{10C_6} = 2^{-\Theta(K/m)}$. Suppose there is an oblivious adversary, and let* Err *denote the number of channel errors the adversary makes. Let* CC *denote the total communication in the entire execution of Algorithm 1), and* EHC *denote the joint number of errors and hash collisions during that execution. Let k be a real number such that $1/\varepsilon^2 \geq k \geq 10C_6$. Then, with probability $1 - p^{\Omega(k\varepsilon|\Pi|)}$, we have that*

$$\mathsf{CC} \leq 200\alpha|\Pi|K \quad \textit{and} \quad \mathsf{EHC} \leq (k+1)\cdot(200\alpha\varepsilon)|\Pi|,$$

or otherwise $\mathsf{Err} > \frac{\varepsilon}{K}\mathsf{CC}$, *where α is the constant multiplying the communication complexity of an iteration in Lemma 2.2.3.*

Overview and A Note on Oblivious Adversaries We will use hash functions with constant collision probability $p = 2^{-\Theta(K/m)}$, which is far higher than the adversarial error rate of ε/K when $K = m$. Despite this, we can still bound the number of hash collisions that occur in the protocol overall by $O(\varepsilon|\Pi|)$. This will follow from the observation that hash collisions are *one-sided*—they can only happen when the transcripts being hashed are different. Since the meeting points protocol lets the parties correct their errors in relatively few steps, there will be few opportunities for hash collisions. A similar approach is taken in [70] in the two-party setting.

Now we give more details. Fix two parties u and v. Note that $\varphi_{u,v}$ roughly measures how many hashes must be passed between the two parties to get back to a consistent transcript, with the property that u and v have a consistent transcript when $\varphi_{u,v} = 0$ (Proposition 2.4.2). The main idea is that the potential function $\varphi_{u,v}$ can only increase by at most a constant during any single exchange of meeting points, even in the presence of errors or hash collisions. Furthermore, in the absence of errors or hash collisions, it decreases by some constant. Finally, if $\varphi_{u,v} = 0$, then it can only increase above 0 if an error is introduced between u and v, because hash collisions do not occur when the transcripts match.

The main approach of this section is to argue that $\varphi_{u,v}$ should not be nonzero too often. For intuition, suppose the adversary starts by making some small number of errors, which makes $\varphi_{u,v}$ equal to some number N. Then the number of iterations in which $\varphi_{u,v}$ is nonzero will be small as long as hash collisions happen infrequently enough that the resulting increase in $\varphi_{u,v}$ does not outweigh the decrease of $\varphi_{u,v}$ in a typical iteration, where a hash collision does not happen. This will follow from the independence of hash collisions across iterations and links of the network.

The hash collisions are independent because the adversary is *oblivious*: they place their errors before seeing the execution of the protocol. What will specifically be useful to us is that they place their errors without knowing the randomness S used to seed the hash function. Since the random seed for the hash h is independently generated for each pair of parties $(u,v) \in E$ and each iteration $i \in [100|\Pi|]$, the events of hash collisions are independent. This lets us bound the number of hash collision with high probability by using a Chernoff bound.

The only property of an oblivious adversary we require here is that the error pattern is independent of the seeds used throughout the protocol. Hence, the results in this section hold for *any* oblivious adversary, regardless of whether the adversary is additive or fixing (see Section 2.1.2 for the definitions of these adversaries). This is because, at the beginning of any iteration i, every partial transcript $T_{u,v}(i)$ is a deterministic function of the inputs to the protocol, the errors that the adversary committed obliviously, and the hash seeds for all links (u,v) and iterations $j = 1,\dots,i-1$. Additionally, in Algorithm 1, the seeds used in iteration i are sampled independently of all of the above.

Proof of Lemma 2.2.11 In analogy with the terminology of dangerous rounds from [70], we define the notion of dangerous triples as follows.

Definition 2.2.12 (Dangerous Triples). *Let i be an iteration of Algorithm 1, and let u and v be parties such that $(u,v) \in E$. Call the triple (i,u,v)* dangerous *if $B_{u,v} > 0$ at the beginning of iteration i.*

Now we state the lemma that we will prove in this section, which will be the main workhorse in proving Lemma 2.2.11.

Lemma 2.2.13. *Let $\varepsilon > 0$ be a sufficiently small constant. Suppose an oblivious adversary, and suppose the hash collision probability of h is p such that $p < \frac{1}{10C_6}$ in Algorithm 11. Let* Err *denote the number of errors the adversary makes,* CC *denote the total communication in Algorithm 1, and let D denote the number of dangerous triples (i,u,v). Let k be a real number such that $1/\varepsilon^2 \geq k \geq 10C_6$. Then with probability $1 - p^{\Omega(k\varepsilon|\Pi|)}$, we have that*

$$\mathsf{CC} \leq 200\alpha|\Pi|K \quad \textit{and} \quad D \leq k \cdot (200\alpha\varepsilon)|\Pi|,$$

or otherwise, $\mathsf{Err} > \frac{\varepsilon}{K}\mathsf{CC}$*, where α is the constant multiplying the communication complexity of an iteration in Lemma 2.2.3.*

Note that a hash collision can only occur between u and v in an iteration $i \in [100|\Pi|]$ when (i, u, v) is a dangerous triple, by definition. Hence, the proof of Lemma 2.2.11 from Lemma 2.2.13 follows easily:

Proof of Lemma 2.2.11. Suppose $\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}$, and recall that $\log(1/p) = \Theta(K/m)$. By Lemma 2.2.13, the communication of Algorithm 1 is upper bounded by $200\alpha|\Pi|K$ with probability $1 - p^{\Omega(k\varepsilon|\Pi|)}$. When this occurs and also $\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}$, this implies that the number of errors is at most $200\alpha\varepsilon|\Pi|$. Furthermore, the number of hash collisions is at most the number of dangerous triples, which is at most $k \cdot (200\alpha\varepsilon)|\Pi|$, again by Lemma 2.2.13. This concludes the proof that EHC is bounded by $(k+1)(200\alpha\varepsilon)|\Pi|$. $\square$

The first part of Lemma 2.2.13 argues that the communication complexity CC is bounded with high probability. First we will argue that if the communication complexity is too large, then the number of dangerous triples must be very large with respect to the number of errors the adversary can introduce. Then, we establish that the number of dangerous triples can only be so large if the fraction of hash collisions in these triples is too large, which happens with low probability.

Lemma 2.2.14. *Consider a run of Algorithm 1. Denote the number of dangerous triples in this run by D, and the number of errors by* Err. *Suppose that the communication complexity in this run satisfies* $\mathsf{CC} > 200\alpha|\Pi|K$, *where α is the constant multiplying the communication in Lemma 2.2.3, and suppose that* $\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}$.

If $\mathsf{Err} > 0$, *then* $D \ge \beta \cdot \mathsf{Err}$, *where* $\beta \overset{\text{def}}{=} \frac{\mathsf{CC}}{3\alpha K \mathsf{Err}} \ge \frac{1}{3\alpha\varepsilon}$. *If* $\mathsf{Err} = 0$, *then $D = 0$ trivially.*

Proof. The final statement follows easily - if there is never any error in the protocol, then there will never be a point at which any pair of transcripts are mismatched.

Now we prove the statement when $\mathsf{Err} > 0$. We start by summing Lemma 2.2.3 over all the $100|\Pi|$ iterations i of Algorithm 1 in order to bound the communication complexity of the entire protocol. Note that $\sum_{i \in [100|\Pi|]} \ell \le \mathsf{EHC}$; recall that EHC is the total number of errors and hash collisions experienced throughout the entire protocol.

$$\begin{aligned} \mathsf{CC} &\le \sum_i \alpha(1+\ell)K \\ &\le \alpha K(100|\Pi| + \mathsf{EHC}) \\ &\le \alpha K\left(100|\Pi| + \frac{\mathsf{CC}\cdot\varepsilon}{K} + D\right) \end{aligned}$$

where the first inequality is Lemma 2.2.3, and in the last inequality we use the fact that

$$\mathsf{EHC} \le \frac{\varepsilon}{K}\mathsf{CC} + D$$

since D upper bounds the number of hash collisions, and that the error rate is bounded by ε/K. Rearranging, we get that

$$\begin{aligned} D &\ge \frac{(1-\alpha\varepsilon)\mathsf{CC} - 100\alpha|\Pi|K}{\alpha K} \\ &\ge \frac{\mathsf{CC}}{3\alpha K} \\ &= \beta\mathsf{Err} \end{aligned}$$

where the second inequality follows from the fact that $\mathsf{CC} > 200\alpha|\Pi|K$ and we take ε sufficiently small so that $\alpha\varepsilon < 1/3$, and the final equality comes from the definition of β. □

Now we proceed with establishing that the probability that D is so large with respect to Err is relatively small. Let $\varphi_{u,v}(i)$ denote the value of $\varphi_{u,v}$ at the beginning of iteration i. Towards proving Lemma 2.2.13, define the random variable $X_{i,u,v}$ for all (i,u,v) such that $\varphi_{u,v}(i) > 0$ as follows:

$$X_{i,u,v} = \begin{cases} 1 & \text{if hash collision occurs between } u \text{ and } v \text{ in iteration } i \\ 0 & \text{otherwise} \end{cases}.$$

Define the process $\psi_{u,v}$ as follows.

Algorithm 4 The process $\psi_{u,v}$

```
$i \leftarrow 1, \psi_{u,v}(1) \leftarrow 0$
**for all** iterations $i$ from 1 to $100|\Pi|$ **do**
    **if** error occurs between $u$ and $v$ during iteration $i$, during any phase **then**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + 6C_6$
    **else if** $\varphi_{u,v}(i) > 0$ **then**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + 5C_6 X_{i,u,v} - 5(1 - X_{i,u,v})$
    **else**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i)$
```

We remark that $\psi_{u,v}$ updates in such a way that it is always a upper bound on $\varphi_{u,v}$. We formalize this below.

Lemma 2.2.15. *For all iterations i in Algorithm 1 and all $(u,v) \in E$, we have that $\psi_{u,v}(i) \geq \varphi_{u,v}(i)$, where $\varphi_{u,v}(i)$ denotes the value of the potential $\varphi_{u,v}$ at the beginning of iteration i.*

Proof. We prove the claim by induction. Clearly it is true for iteration $i = 1$. Assume now that it is true for a certain i. We will show that it is also true for iteration $i+1$.

If $\varphi_{u,v}(i+1) = 0$, then the claim follows since in this case $\psi_{u,v}(i+1) \geq \psi_{u,v}(i)$, and $\psi_{u,v}(i) \geq \varphi(i)_{u,v} \geq 0$ by induction and since $\varphi_{u,v}(i)$ is non-negative (Proposition 2.4.2).

Suppose there is an error between u and v in iteration i. We know that $\varphi_{u,v}$ increases by at most $6C_6$ regardless of the number of errors or hash collisions in the entire iteration. This follows from Lemma 2.4.6 and Claim 2.4.1: Lemma 2.4.6 shows that $\varphi_{u,v}$ can increase by at most $5C_6$ in the Meeting Points phase, and Claim 2.4.1 shows that $\varphi_{u,v}$ can increase by at most $2C_3$ in all the other phases combined, so $C_6 \geq 2C_3$ yields the desired result. So we conclude that $\psi_{u,v}(i+1) \geq \varphi_{u,v}(i+1)$.

Now suppose that $\varphi_{u,v}(i+1) > 0$ and there is no error between u and v in iteration i. Then we must have that $\varphi_{u,v}(i) > 0$, since if $\varphi_{u,v}(i) = 0$, we would have $B_{u,v}(i) = 0$ and, therefore there is no error or hash collision in iteration i between u and v and hence the

$\varphi_{u,v}$ cannot increase (Proposition 2.4.4, Lemma 2.4.6 for the Meeting Points phase, and Claim 2.4.1 for Flag Passing, Simulation, and Rewind phases). Furthermore, there can only be a hash collision at iteration i between u and v if $\varphi_{u,v}(i) > 0$, which follows from Proposition 2.4.2 and the observation that hash collisions can only happen if $B_{u,v}(i) > 0$ after iteration i. Hence, $\psi_{u,v}(i+1) = \psi_{u,v}(i)+5C_6$ whenever there is a hash collision between u and v, and $\psi_{u,v}(i+1) = \psi_{u,v}(i) - 5$ whenever there is no hash collision between u and v. Since $\varphi_{u,v}$ increases by at most $5C_6$ in the presence of a hash collision and decreases by at least 5 in the absence of one (Lemma 2.4.6), we conclude the proof. □

Let D^* denote the set of triples (i,u,v) such that $\psi_{u,v}(i) > 0$. Then we claim that $D \le |\mathsf{D}^*|$. This follows from the fact that $\psi_{u,v}(i) \ge B_{u,v}(i)$ for all i (Lemma 2.2.15 and Proposition 2.4.2). Our goal will be to prove that $|\mathsf{D}^*|$ is not much larger than Err, with high probability. Our strategy will be to use the fact that $\psi_{u,v}(i)$ is always nonnegative (again, an application of Lemma 2.2.15 and Proposition 2.4.2) and that the adversary cannot make too many errors, to argue that $\sum_{(i,u,v)\in\mathsf{D}^*} X_{i,u,v}$ must be bounded below by something relatively large. If there are many random variables $X_{i,u,v}$, then a Chernoff bound will let us argue that $\sum_{(i,u,v)\in\mathsf{D}^*} X_{i,u,v}$ should not be this large with high probability. However, the communication in the protocol is not a priori bounded[12]: if there are many hash collisions or errors, then the parties might communicate more, which creates more budget for errors. So our first step is to bound the number of errors that the adversary can commit.

Lemma 2.2.16. *Suppose that the adversary is oblivious, and commits* Err *errors. Denote the hash collision probability in Algorithm 1 as p, and suppose that $p < 1/30C_6$. Then,*

$$\mathbb{P}\left[\mathsf{Err} \le 200\alpha\varepsilon|\Pi| \bigvee \mathsf{Err} > \frac{\varepsilon}{K}\mathsf{CC}\right] \ge 1 - p^{\Omega(|\Pi|/\varepsilon)},$$

where the probability above is over the random seeds used for hashing throughout the protocol.

Note that while the number of errors that an oblivious, additive adversary commits is fixed ahead of time, the communication in Algorithm 1 (denoted CC) is dependent on

[12]Except trivially, by the number of rounds of Algorithm 1 times m, but this can be a factor m more than $CC(\Pi)$.

the seeds, which is why the event in Lemma 2.2.16 is not deterministic. Note that for an oblivious fixing adversary, the number of errors they commit is also a random variable, as it depends on the communication pattern of the protocol, since e.g. fixing a link to communicate a 1 in a certain round does not constitute an error if indeed there is a 1 sent in that round. In both cases, the probability is over the seeds sampled for the hashes.

Proof of Lemma 2.2.16. Suppose $\mathsf{Err} > 200\alpha\varepsilon|\Pi|$ and $\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}$. This implies that $\mathsf{CC} > 200\alpha|\Pi|K$, and so applying Lemma 2.2.14 we see that $D > 1/(3\alpha\varepsilon)\mathsf{Err}$. As we noted previously, $|\mathsf{D}^*| \ge D$ (consequence of Lemma 2.2.15 and argument above), and so it suffices to bound the probability that $|\mathsf{D}^*|$ exceeds $1/(3\alpha\varepsilon)\mathsf{Err}$. By taking ε small enough with respect to α and C_6, we can safely assume that $3\alpha\varepsilon < 1/(10C_6)$, and therefore we can apply Lemma 2.2.17, which tells us that the probability of this is at most $\exp(-\Omega(\mathsf{Err}\log(1/p)/\varepsilon))$. □

Lemma 2.2.17. *Suppose the adversary is oblivious, and let* Err *be the number of errors committed during a given instance of the protocol. Let p be the hash collision probability of h, and assume that $p < 1/30C_6$. Let k be some real number such that $k \ge 10C_6$. Let* D^* *denote the set of triples (i, u, v) such that $\psi_{u,v}(i) > 0$. Then,*

$$\mathbb{P}[|\mathsf{D}^*| > k\mathsf{Err}] < p^{\Omega(k\mathsf{Err})},$$

where the Ω hides constants on the order of $1/C_6$.

Proof. Consider running the protocol after fixing the adversary's errors with uniformly random seeds for the hashes. Assume that $|\mathsf{D}^*| > k\mathsf{Err}$, and let $\widetilde{\mathsf{D}^*} \subseteq \mathsf{D}^*$ be the subset of triples where no error occurs. Since the number of errors is Err, we get that $|\widetilde{\mathsf{D}^*}| \ge (k-1)\mathsf{Err}$. Define

$$\psi \overset{\text{def}}{=} \sum_{(u,v)\in E} \psi_{u,v}.$$

We will argue that the fraction of hash collisions required to be in the $|\widetilde{\mathsf{D}^*}|$ (possibly) dangerous triples is far too large.

We know that ψ is always nonnegative by design, since each of the $\psi_{u,v}$'s are nonnegative. Consider $\psi(100|\Pi|+1)$, that is, the value of ψ immediately after the final $100|\Pi|$-th iteration of Algorithm 1. Then, recalling the definition of $\psi_{u,v}$, we can upper bound $\psi(100|\Pi|+1)$ as follows.

$$0 \leq \psi(100|\Pi|+1) \leq 6C_6 \cdot \mathsf{Err} + \sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} (5C_6 X_{i,u,v} - 5(1 - X_{i,u,v})). \tag{2.7}$$

Hence, we can conclude that

$$\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} 5C_6 X_{i,u,v} - 5(1 - X_{i,u,v}) \geq -6C_6 \cdot \mathsf{Err}. \tag{2.8}$$

Now we claim that $\sum X_{i,u,v} \geq \frac{4}{5C_6+5}|\widetilde{\mathsf{D}^*}|$.

$$\begin{aligned}
\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} (5C_6 X_{i,u,v} - 5(1 - X_{i,u,v})) &\geq -6C_6 \cdot \mathsf{Err} \\
(5C_6+5)\left(\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v}\right) - 5|\widetilde{\mathsf{D}^*}| &\geq -6C_6 \cdot \mathsf{Err} \\
\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} &\geq \frac{4}{5C_6+5}|\widetilde{\mathsf{D}^*}|,
\end{aligned}$$

where in the last line we use the fact that $|\widetilde{\mathsf{D}^*}| \geq (k-1)\cdot \mathsf{Err} \geq 6C_6 \cdot \mathsf{Err}$.

Technically, some of the $X_{i,u,v}$ are deterministically 0, which occurs when $\varphi_{u,v}(i) > 0$ but $B_{u,v} = 0$. The event of $X_{i,u,v}$ being deterministically 0 is also influenced by previous hash collisions in the network, as these play a role in making the transcripts differ. We will ignore this fact wlog, since making all of the $|S'|$ triples have some probability of have collision (rather than just some of them) can only increase $\sum X_{i,u,v}$, and we will still be able to show that $\sum X_{i,u,v}$ is small. So we will assume that all the random variables $X_{i,u,v}$ are i.i.d Ber(p).

Now we apply a Chernoff bound. By taking $p < 1/(10C_6)$, we note that

$$\mathbb{P}\left[\sum_{\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq \frac{4}{5C_6+5}|\widetilde{\mathsf{D}^*}|\right] \leq \mathbb{P}\left[\frac{1}{|\widetilde{\mathsf{D}^*}|}\sum X_{i,u,v} \geq p + \frac{3}{5C_6+5}\right].$$

We bound this probability by $p^{\Omega(|\widetilde{\mathsf{D}^*}|)}$ in Claim 2.2.18, where the Ω hides terms on the order of $1/C_6$. We finish the proof by recalling that $|\widetilde{\mathsf{D}^*}| \geq (k-1)\cdot \mathsf{Err}$. □

Claim 2.2.18. *Fix a constant $\gamma > 0$. Suppose that $\{X_{i,u,v}\}_{(i,u,v)\in\widetilde{\mathsf{D}^*}}$ are i.i.d Ber(p) random variables such that p is sufficiently smaller than γ. Then*

$$\mathbb{P}\left[\frac{1}{|\widetilde{\mathsf{D}^*}|}\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq p+\gamma\right] \leq p^{\Omega(|\widetilde{\mathsf{D}^*}|)}$$

where the Ω hides constants on the order of γ.

Proof of Claim. Recall from Theorem 1.2.1 that

$$\mathbb{P}\left[\frac{1}{|\widetilde{\mathsf{D}^*}|}\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq p+\gamma\right] \leq \exp\left(-D\left(p+\gamma \,\middle\|\, p\right)\cdot|\widetilde{\mathsf{D}^*}|\right)$$

We claim that $D(p+\gamma\|p) \geq \Omega(\log(1/p))$.

$$\begin{aligned} D(p+\gamma\|p) &= (p+\gamma)\ln\left(\frac{\gamma}{p}\right) + (1-p-\gamma)\ln\left(\frac{1-p-\gamma}{1-p}\right) \\ &\geq \gamma\ln\left(\frac{\gamma}{p}\right) + (1-p-\gamma)\left(\frac{-2\gamma}{1-p}\right) \\ &\geq \gamma\ln\left(\frac{\gamma}{p}\right) - 2\gamma \\ &\geq \gamma\left(\ln\left(\frac{\gamma}{p}\right) - 2\right) \end{aligned}$$

where in the second line we use the inequality that $-x \leq \ln(1-x/2)$ for $x \in (0,1)$. □

From the above lemmas, the proof of Lemma 2.2.13 follows easily.

Proof of Lemma 2.2.13. By Lemma 2.2.16, the probability of having more than $200\alpha\varepsilon|\Pi|$ errors in the protocol while also having $\mathsf{Err} \leq \frac{\varepsilon}{K}\mathsf{CC}$ is at most $p^{\Omega(|\Pi|/\varepsilon)}$. The probability that the number of errors is smaller than $200\alpha\varepsilon|\Pi|$ and simultaneously the number of dangerous triples exceeds $k \cdot (200\alpha\varepsilon)|\Pi|$ is at most $p^{\Omega(k\varepsilon|\Pi|)}$, by Lemma 2.2.17[13]. Hence, the union of these two events has probability at most

$$p^{\Omega(k\varepsilon|\Pi|)},$$

which is the only place where we use that $k \leq 1/\varepsilon^2$. □

Now that we have bounded the communication and number of hash collisions, we are ready to prove Theorem 2.2.2.

2.2.6 Completing the proof of Theorem 2.2.2

Recall that we set $K := m$ in InitializeState (Algorithm 2). Throughout this analysis, we assume that the number of errors that the adversary commits is Err such that $\mathsf{Err} \leq (\varepsilon/m)\mathsf{CC}$, as this is what is implied by the adversary having rate ε/m.

Lemma 2.2.3 shows that in every iteration of Algorithm 1, ϕ increases by at least $K = m$. Hence, after at the end of the simulation after $100|\Pi|$ iterations, we know that $\phi \geq 100|\Pi|m$.

Lemma 2.2.11 shows that the total communication during the protocol is bounded by $\mathsf{CC} \leq 200\alpha|\Pi|m$ with probability at least $1 - p^{\Omega(|\Pi|/\varepsilon)}$, by invoking it with $k = 1/\varepsilon^2$. Since we take p and α to be constants, this just says that $\mathsf{CC} = O(|\Pi|K) = O(\mathsf{CC}(\Pi))$ with probability $1 - \exp(-\Omega(|\Pi|/\varepsilon))$. Hence, we will assume that the $\mathsf{CC} = O(|\Pi|K)$ for the remainder of the proof.

We conclude the argument by establishing that, when the protocol ends and $\phi \geq 100|\Pi|m$, the parties are done simulating the initial protocol correctly with high probability.

[13]Note that the k in the statement of Lemma 2.2.13 and the k with which we invoke Lemma 2.2.17 here are different - technically, we invoke Lemma 2.2.17 with its k set to $k(200\alpha\varepsilon|\Pi|)/\mathsf{Err}$. We settle for this abuse of notation because k plays the same role conceptually in Lemma 2.2.13 and Lemma 2.2.17.

We do this by appealing to the following claim, written for general K, and apply it with $K = m$ to finish the proof.

Claim 2.2.19. *Suppose that $\phi \geq 100|\Pi|K$. Then with probability $1 - \exp(-\Omega(|\Pi|))$, the parties are done simulating the underlying protocol Π correctly.*

Proof. Recall our definition of the potential ϕ:

$$\phi = \sum_{(u,v)\in E} ((K/m)G_{u,v} - K \cdot \varphi_{u,v}) - C_1 K B^* + C_7 K \cdot \mathsf{EHC}.$$

Fix ε^* to be a constant such that $\varepsilon^* \leq 1/(4200 C_6 C_7 \alpha)$, and take ε to be sufficiently smaller than ε^*. Since $\varepsilon^* \geq \varepsilon$, know that $\mathsf{EHC} \leq 4200 C_6 \alpha \varepsilon^* |\Pi|$ with probability at most $1 - \exp(-\Omega(\varepsilon^* |\Pi|))$. This follows from Lemma 2.2.11, taking $k + 1 := (4200 C_6 \alpha \varepsilon^* / \varepsilon) \cdot (|\Pi| m / \mathsf{CC})$, and noting that $|\Pi| m / \mathsf{CC}$ is a constant, so taking ε sufficiently small makes k be large enough for the lemma to apply.

Since ε^* is sufficiently small, we get that

$$C_7 K \cdot \mathsf{EHC} \leq C_7 K \cdot 4200 C_6 \alpha \varepsilon^* |\Pi| \leq |\Pi| K.$$

Furthermore, the term $\sum -K \cdot \varphi_{u,v}$ is nonpositive, due to the fact that $\varphi_{u,v}$ is non-negative (Proposition 2.4.2). Therefore, we get that

$$\sum_{(u,v)\in E} \frac{K}{m} G_{u,v} - C_1 K B^* \geq 99|\Pi|K. \tag{2.9}$$

Recall that $B^* = H^* - G^* \geq \max_{(u,v)\in E}(G_{u,v}) - \min_{(u,v)\in E}(G_{u,v})$. By plugging this into

Eq. (2.9), and recalling that $C_1 \geq 2$, we get

$$\begin{aligned}
99|\Pi|K &\leq \sum_{(u,v)\in E} \frac{K}{m} G_{u,v} - C_1 K B^* \\
&\leq K \max_{(u,v)\in E}(G_{u,v}) - C_1 K \left(\max_{(u,v)\in E}(G_{u,v}) - \min_{(u,v)\in E}(G_{u,v}) \right) \\
&\leq K \max_{(u,v)\in E}(G_{u,v}) - K \left(\max_{(u,v)\in E}(G_{u,v}) - \min_{(u,v)\in E}(G_{u,v}) \right) \\
&= K \min_{(u,v)\in E}(G_{u,v})
\end{aligned}$$

Hence, we conclude that $\min_{(u,v)\in E}(G_{u,v}) \geq 99|\Pi|$. Therefore, each pair of parties have simulated Π correctly for at least $|\Pi|$ chunks, which suffices to compute Π correctly.

Finally, we recall that this all happens with probability $1 - \exp(-\Omega(\varepsilon^*|\Pi|))$. Since ε^* is a fixed constant in terms of α, C_6, and C_7, we can absorb it into the Ω, and conclude that the above happens with probability $1 - \exp(-\Omega(|\Pi|))$. □

2.3 Part II: Non-Oblivious Noise

2.3.1 Introduction

Our Contributions and Comparison to Previous Work

We design coding schemes for adversarial errors in the same communication model as [85] and Section 2.2; namely, where parties may or may not speak in a given round, and the underlying protocol is only assumed to have a fixed speaking order, which is assumed by all known schemes in the non-fully-utilized setting. As in Section 2.2, our coding schemes are for networks with an arbitrary topology, are computationally efficient, and feature a constant rate (i.e., a constant multiplicative blowup in communication). Our schemes are randomized, however they *do not* assume the parties pre-share any randomness. The schemes succeed with high probability assuming an all-powerful adversary that may insert, delete, and substitute bits up to a restricted fraction of the communication.

Our first coding scheme (Algorithm A) achieves a constant rate, and is resilient to ε/m-fraction of adversarial insertion, deletion, and substitution noise. This scheme assumes a limited type of noise, i.e., an *oblivious adversary* whose corruptions are predetermined and are independent of the parties' randomness. In our second coding scheme (Algorithm B), we remove the restriction of the obliviousness of the adversary, at the price of being resilient to $\varepsilon/(m \log m)$-fraction of errors.

Finally, if we remove only the assumption that the adversary is oblivious, but retain the assumption that the parties pre-share a long random string, then we obtain a scheme that is resilient to a higher level of noise, namely, to $\varepsilon/m \log\log m$-fraction of insertion and deletion errors (Algorithm C). See Table 2.1 for a comparison of our new results (Algorithms A, B, and C) with the state of the art. See Section 2.2 for further discussion on related work.

More formally, our result in this section is two-fold. First, we assume that the adversary is oblivious, i.e., the noise is predetermined and is independent of the randomness used by the parties.

scheme	topology	noise level	noise type	rate	efficient
RS94	arbitrary		BSC_ε	$1/O(\log(d+1))$	
ABGEH16	clique		BSC_ε	$\Theta(1)$	Yes
HS16	arbitrary	$O(1/m)$	substitution	$\Theta(1)$	
HS16	arbitrary	$O(1/n)$	substitution	$1/O(m\log(n)/n)$	
JKL15	star	$O(1/m)$	substitution	$\Theta(1)$	
Algorithm A	arbitrary	$O(1/m)$	*oblivious* insertions and deletions	$\Theta(1)$	Yes
Algorithm B	arbitrary	$O(1/m\log m)$	insertions and deletions	$\Theta(1)$	Yes
pre-shared randomness:					
Algorithm C	arbitrary	$O(1/m\log\log m)$	insertions and deletions	$\Theta(1)$	Yes

Table 2.1: *Interactive coding schemes in the multiparty setting. The first four are in the* fully-utilized *model (each channel must be utilized in every round), while [85] and all algorithms in this paper are in the non-fully-utilized model. Recall that n is the number of parties, and m is the number of (bi-directional) communication links in the network.*

Theorem 2.3.1 (Oblivious noise, informal). *Let $G = (V,E)$ be an arbitrary synchronous network with $n = |V|$ nodes and $m = |E|$ links. For any noiseless protocol Π over G with a predetermined order of speaking, and for a sufficiently small constant ε, there exists a computationally efficient coding scheme that simulates Π over a noisy network G. The simulated protocol is robust to adversarial insertion, deletion, and substitution noise, assuming at most ε/m-fraction of the communication is corrupted. The simulated protocol communicates $O(\mathsf{CC}(\Pi))$ bits, and succeeds with probability at least* $1-\exp(-\mathsf{CC}(\Pi)/m)$*, assuming the noise is oblivious.*

Next, we remove the restriction to oblivious noise. Namely, we consider adversaries that may adaptively decide which transmissions to corrupt according to the observed transcript, as well as the parties inputs (however, the noise is unaware of any private coin-tossing a party may perform later in the protocol). In this case we still obtain an efficient coding scheme with a constant rate, albeit, with slightly smaller noise resilience.

Theorem 2.3.2 (Non-oblivious noise, informal). *Let $G = (V,E)$ be an arbitrary synchronous network with $n = |V|$ nodes and $m = |E|$ links. For any noiseless protocol Π over G with a predetermined order of speaking, and for a sufficiently small constant ε, there exists a com-*

putationally efficient coding scheme that simulates Π *over a noisy network* G. *The simulated protocol is robust to adversarial insertion, deletion, and substituition noise, assuming at most* $(\varepsilon/m\log m)$*-fraction of the communication is corrupted. The simulated protocol communicates* $O(\mathsf{CC}(\Pi))$ *bits, and succeeds with probability at least* $1-\exp(-\mathsf{CC}(\Pi)/m)$.

In Section 2.5 we consider the case where the adversarial channel is non-oblivious, however, the parties pre-share a long random string. In this case we show a coding scheme (Algorithm C) that is resilient to a somewhat higher noise level of $\varepsilon/m\log\log m$-fraction of insertion and deletion noise, while still incurring a constant blowup in the communication. See Section 2.5 for the complete details.

We now describe how we modify Algorithm 1 from Section 2.2.

Removing the Common Random String assumption. Recall that the parties generate hashes of their partial simulated transcripts in the beginning of any iteration, in the consistency check phase. The hash function is chosen at random by sampling a (uniformly) random seed. These seeds together constitute the CRS.

A basic idea for removing the CRS is to simply have each pair of parties sample this seed and send it across their shared link. However, the length of the seed is similar to the length of the information we wish to hash (see Section 2.3.2 for further details on the hash function we use). Hence, sending it will increase the communication complexity by a quadratic factor. Instead, we use an idea from previous work [61, 25, 70, 62]: the parties communicate a short random string S which serves as a seed that generates a longer δ-biased string. This is done via a well-known technique by Naor and Naor [122], or Alon et al. [5]. This δ-biased randomness is used instead of the shared randomness above.

Intuitively, δ-biased randomness suffices, since the hash function is linear, and thus hashing with a δ-biased seed behaves "close" to hashing with a truly uniform seed. However, this holds only when the input to the hash function is independent of the δ-biased seed. Unfortunately, in our setting, the input to the hash function may depend on the δ-biased seed. Clearly, if the adversary is non-oblivious, they can look at the the δ-biased seed and choose their errors as a function of it, so the partial transcripts may be correlated with the seeds used to hash them. More problematic is that even if the adversary is *oblivious*, the partial transcripts may depend on the seeds being used to hash them. Indeed, this is because the seeds are taken from a long δ-biased seed, and so seeds being used in the current round are correlated with previously-used seeds. Specifically, the previous seeds determined whether there were hash collisions or not, which determines the state of the current partial transcript.

We circumvent this issue in a way similar to [70, 24], by showing that a given pattern of hash collisions determines the (partial) transcripts throughout the protocol. Once the transcripts are fixed, the hashes output behave similarly to the case of uniform randomness, up to a statistical difference of at most δ. Therefore, if δ is small enough, we can prove robustness for all possible fixed patterns of hash collisions. Then, we can union bound over all the possible patterns of hash collisions, which bounds the failure probability for the δ-biased case by the failure of the uniform case plus an error term that depends on δ and the cardinality of the different hash patterns. One cannot take δ to be too small, since generating the δ-biased string requires communicating $\approx O(m \log(1/\delta))$ bits, hence the smaller δ gets, the more communication we need. Luckily, we can set δ to be a sufficiently small constant so that the probability of failure is still exponentially small even after the union bound, while keeping a constant rate in the communication.

Dealing with a non-oblivious adversary. Next, we consider the case where the adversary is non-oblivious. In particular, we assume that the adversary knows all the randomness of the parties. However, the adversary is still oblivious to any private randomness that a party may have.

Haeupler [70] dealt with this same issue in the two-party setting by simply applying a union bound over all the possible oblivious attacks. Unfortunately, as opposed to the two-party case of [70], knowledge of the hash seeds in the multi-party setting gives the adversary too many options for attacks, and the proof fails spectacularly.

To see why this is, fix an edge (u, v) in the network, and suppose for simplicity that we use truly random functions for our hashes. Since we use constant-length hash outputs[14] and the adversary knows the random seeds ahead of time, with constant probability over the random seed, the adversary can corrupt the simulation on the link (u,v) such that a hash-collision is guaranteed in the following consistency-check phase. In this case, the parties will continue to simulate (incorrectly) since they are not aware of the mismatch in their partial transcripts. However, note that with some constant probability, this mismatch will not be caught in the next consistency-check as well! Generalizing the above reasoning, with probability $1/m$, the adversary is able to force hash-collisions on the link (u,v) for $\Theta(\log m)$ consecutive phases, leading to $\Theta(m \log m)$ wasted communication. Since there are m links for the adversary to choose from when making an error, with high probability there exists a link on which such an attack is possible, and the adversary who knows the random seeds, the inputs, and the protocol will be able to conduct this attack on it. Therefore, the non-oblivious adversary can create $\Theta(m \log m)$ wasted communication with just a single error.

To overcome this issue, we increase the length of the hash output, so it is no longer constant but rather $\Theta(\log m)$. Then the hash collision probability drops to $1/\text{poly}(m)$, and the union bound yields the desired outcome. However, the length of each hash increased from a constant to $\Theta(\log m)$, and the rate of the coding scheme is no longer constant (recall that each iteration in Algorithm 1 from Section 2.2 takes $\Theta(m)$ communication). In order to retain a constant rate, we simulate the protocol in larger chunks. Namely, each phase (i)-(iv) now consists of $\Theta(m \log m)$ bits. In particular, instead of sending a hash every $\Theta(m)$ bits of communication of Π, parties exchange hashes of length $\Theta(\log m)$ every $\Theta(m \log m)$ bits of simulation, thus increasing the overall communication by just a constant factor. However, as a result, we are resilient to at most $\varepsilon/(m \log m)$ fraction of adversarial error.

[14]In order to obtain a constant rate, the length of the hash output must be constant, see Section 2.2.

2.3.2 Preliminaries

In this section we set some notations and provide lemmas that we will use in the rest of this section.

Codes

We use a standard binary error-correction code with constant rate and constant distance, which has efficient encoding and decoding procedures. Such codes can be constructed by concatenating Reed-Solomon codes with binary linear random codes, or by employing the near-linear codes by Guruswami and Indyk [66].

Theorem 2.3.3. *For every $0 < \rho < 1$ there exists $\delta > 0$ s.t. the following holds for all sufficiently large n. There exists a binary linear code $C : \{0,1\}^k \to \{0,1\}^n$ with rate $k/n \geq \rho$ and relative distance at least δ. Furthermore, C can be encoded and decoded from up to $\delta/2$ fraction of bit-flips in polynomial time in n.*

Hash functions, δ-biased strings

Usually, the hash function from Definition 2.1.2 is seeded with a uniform string. In order to reduce the amount of randomness needed, we use δ-biased random strings, which are close enough to uniform (for our needs), yet can be constructed from much shorter (uniform) seeds, via a result by Naor and Naor [122].

Definition 2.3.4 (δ-bias)**.** *Fix $\delta > 0$. A distribution $\mathcal{D}$ over $\mathbb{F}_2^n$ is δ-biased if for any $v \in \mathbb{F}_2^n \backslash \{0^n\}$, we have that*

$$\left| \mathbb{P}_{x \sim \mathcal{D}} \left[\sum_{i=1}^{n} v_i x_i = 0 \right] - 1/2 \right| \leq \delta.$$

Lemma 2.3.5 ([122, 5])**.** *There is a constant $c \in \mathbb{N}$ and an efficiently computable function $G : \{0,1\}^* \to \{0,1\}^*$ such that the following holds. Fix $\varepsilon > 0$. For any size k and a uniformly random string $S \in \{0,1\}^{c \cdot (k + \log(1/\varepsilon))}$, we have that $G(S) \in \{0,1\}^{2^k}$ is a 2ε-biased distribution over bit strings of length of 2^k.*

Finally, we appeal to the following Lemma from [70] (which is in turn based on [122]) that connects the behaviour of hash function seeded with δ-biased string to hashes seeded with uniformly random string *as long as their input is fixed and independent of the seed.*

Lemma 2.3.6 (Lemma 6.3 from [70]). *Fix positive integers k, τ, and L. Consider k pairs of binary strings $(x_1, y_1), \dots (x_k, y_k)$ where each string has length at most L. Let $\{h_Q\}_{Q \in \{0,1\}^{2\tau L}}$ be an inner product hash family with input length $\leq L$, output length τ, and seed length $2\tau L$. Let $S = (s^{(1)}, \dots, s^{(k)})$ be a random seed of length $k \cdot 2\tau L$.*

1. *If S is drawn from a uniform distribution over $\{0,1\}^{2\tau L \cdot k}$, then for each $i \in [k]$,*

$$\mathbb{P}_{s^{(i)}}[h_{s^{(i)}}(x_i) = h_{s^{(i)}}(y_i)] = 2^{-\tau}$$

   if $x_i \neq y_i$. Note that, trivially, $\mathbb{P}_{s^{(i)}}[h_{s^{(i)}}(x_i) = h_{s^{(i)}}(y_i)] = 1$ when $x_i = y_i$. Furthermore, for each $i \in [k]$ the events of hash collisions are independent.

2. *If S is drawn from a δ-biased distribution over $\{0,1\}^{2\tau L \cdot k}$, then it holds that the distribution*

$$\left(1_{h_{s^{(1)}}(x_1) = h_{s^{(1)}}(y_1)}, \dots, 1_{h_{s^{(k)}}(x_k) = h_{s^{(k)}}(y_k)}\right)$$

   is δ-close to the case where S is uniformly random.

Recall that δ-closeness means that the statistical difference between two distributions is bounded by δ.

Definition 2.3.7 (δ-closeness). *We say that distributions P, Q over the probability space Ω are δ-close if*

$$\sup_{A \subseteq \Omega} |P(A) - Q(A)| \leq \delta.$$

The above is equivalent to having $\frac{1}{2}\sum_{\omega \in \Omega} |P(\omega) - Q(\omega)| \leq \delta$.

Note that as a corollary of Lemma 2.3.6, by setting k to 1, τ to a constant, and using δ-biased randomness with $\delta = 2^{-\tau}$, we can get the hash functions with seed length that is logarithmic in the size of their input, by seeding the inner product hash function with δ-biased seeds. This corollary was noted in previous work, e.g. in Naor and Naor [122].

Corollary 2.3.8. *There is a hash function family $\{h_Q\}_{Q \in \{0,1\}^s}$ with input length $\leq L$, output length τ, and seed length $s = \Theta(\log L + \tau)$ such that, given any pair of inputs x and y such $x \neq y$, we have that*

$$\mathbb{P}_Q[h_Q(x) = h_Q(y)] \leq 2 \cdot 2^{-\tau}$$

2.3.3 Coding Scheme for Oblivious Noise without Pre-shared Randomness

In this section we modify our protocol from Section 2.2 to not require a (preshared) common random string (CRS) between the parties. This will involve the parties generating randomness privately and sharing this randomness with their neighbors. In this section we will still assume that the adversary is oblivious, which in particular means that the adversary cannot plan his errors after seeing the random strings communicated in the network.

Let us recall the scheme of Section 2.2. Recall that this protocol works in iterations, where each iteration consists of 4 phases: (i) consistency check, (ii) flag passing, (iii) simulation, and (iv) rewind. In each consistency check, every two neighboring parties u and v exchange hashes of $T_{u,v}$ (and $T_{v,u}$) to verify that their transcripts so far are consistent. If their hashes are inconsistent, they initiate a meeting points mechanism to find the last point they were in agreement. In addition, in phase (ii), they send a flag to all the parties in the network indicating that the simulation should be temporarily stopped. If their hashes are consistent and they do not receive a flag from other parties, then they simulate in phase (iii).

Note that the pairwise transcripts $T_{u,v}$ are extremely long, potentially $O(|\Pi|K)$ bits, where $|\Pi|$ denotes the number of chunks in the underlying protocol and K is such that the amount of communication in any single chunk is $O(K)$. In particular, in our first protocol against oblivious adversaries (Algorithm A) we set $K = m$, in our second protocol against any (adaptive) adversary we set $K = m \log m$ (Algorithm B), and in our third protocol (Algorithm C, in Appendix 2.5), we set $K = m \log\log m$. This means that if we use the inner product hash function of Definition 2.1.2 and assume a uniformly random seed, the seed's length will be $O(|\Pi|K)$ bits. Since the parties do not pre-share randomness, they must communicate these seeds in order to agree on the seeds to be used. However, if they all send each other $|\Pi|$ uniformly random seeds each of length $\Omega(|\Pi|K)$, this would cost $\Omega(|\Pi| \cdot m \cdot \mathsf{CC}(\Pi))$ in communication throughout the protocol, and the rate of the coding scheme would approach zero.

Our way around this, much like the solution employed by previous work [61, 25, 70], is to use δ-biased seeds for the inner product hash function. Specifically, recall that the lemma of Naor and Naor [122] (Lemma 2.3.5) states that we can efficiently generate δ-biased distributions over strings of length ℓ using just $\Theta(\log(1/\delta) + \log(\ell))$ bits of uniform randomness. Hence, the parties can share small amounts of uniform randomness that they will expand to a common δ-biased random seed, which they can then use for the consistency checks.

The coding scheme construction

We now define our first interactive coding scheme, which does not assume a CRS, but does assume that the adversary is oblivious, which we call *Algorithm A*. It is resilient to a fraction ε/m of adversarial error. In Algorithm 1 we recall the blueprint of the scheme from Section 2.2. The blueprint of Algorithm A is identical to Algorithm 1 from Section 2.2, the only difference being the implementation of the subroutine InitializeState(), which is replaced with Algorithm 6. This new subroutine initiates a randomness exchange routine (Algorithm 7), where the parties exchange a small amount of uniform randomness in order to generate a larger (shared) δ-biased seed, which replaces the pre-shared uniform random string assumed in Algorithm 1. The other subroutines in Algorithm A, namely, the flag-passing mechanism (Algorithm 3) and the meeting-points mechanism (Algorithm 11), remain as defined for Algorithm 1 in Section 2.2. We now give the blueprint of Section 2.2

(Algorithm 1) and the subroutines for Algorithm A. We repeat the flag passing algorithm for ease of reading, even though it is identical to what is presented in Section 2.2.

The randomness exchange procedure is depicted in Algorithm 7. On every link (u, v), one of the parties u (chosen according to some global total ordering of the parties) uniformly samples a seed L, encodes it against errors, and sends it to v. Then, both parties expand R to a larger δ-biased string $S_{u,v}$. Note that v decodes the communication from u and gets a seed L' which may differ from L due to noise. In this case the string $S_{u,v}$ generated by u will differ from $S_{v,u}$ generated by v. However, we will argue that, with high probability, the adversary will not have enough noise budget to corrupt any $S_{v,u}$.

We note that the randomness exchange uses a standard error correcting code in order to protect the seeds from being shared incorrectly unless the adversary commits a large number of errors. We do not need an ins/del code because that the rounds where the seeds are shared are fully utilized, and so deletions are equivalent to erasures.

We now give the FlagPassing procedure (Algorithm 3), which is identical to the one in Section 2.2. This part assumes that all the parties share a spanning tree of the network graph, rooted at some node ρ. If we don't assume that all the parties know the topology of the network, then such a tree can be constructed in a resilient manner using the techniques in [32].

Algorithm 5 (Algorithm 1 from Section 2.2) A noise-resilient simulation of Π (for party u)

Let $T_{u,v}$ denote the partial, pairwise transcript between u and v *according to* u, and let $|T_{u,v}|$ denote the number of *chunks* simulated so far in $T_{u,v}$.

```
1: InitializeState( )

2: for $i = 1$ to $100|\Pi|$ do
3:     for all $v \in \mathcal{N}(u)$ in parallel do                                   ▷ meeting points
4:         $status_{u,v} \leftarrow$ MeetingPoints$(u, v, S_{i,u,v})$
5:     $minChunk \leftarrow \min_{v\in\mathcal{N}(u)} |T_{u,v}|$
6:     if exists $v$ such that $status_{u,v}$ = "meeting points" then
7:         $status_u \leftarrow 0$
8:     else if exists $v$ such that $|T_{u,v}| > minChunk$ then
9:         $status_u \leftarrow 0$
10:    else
11:        $status_u \leftarrow 1$

12:    $netCorrect_u \leftarrow$ FlagPassing$(u, status_u)$                                ▷ flag passing

13:    if $netCorrect_u = 1$ then                                                      ▷ simulation
14:        Listen for one round.
15:        Simulate chunk $|T_{u,v}| + 1$ with each party $v \in \mathcal{N}(u)$ from whom we
           have not received $\perp$ at the first round. The simulation is based on the
           partial transcript $T_{u,w}$ for each $w \in \mathcal{N}(u)$, as well as the input to $u$.
16:        If the above step took less than $5K$ rounds, wait until $5K$ rounds have passed
17:        if received no $\perp$'s in Line 14 in this iteration then
18:            $minChunk \leftarrow minChunk + 1$
19:    else
20:        Send a single $\perp$ to each neighbor, and wait $5K$ rounds.

21:    for round $r = 1$ to $n$ do                                                     ▷ rewind
22:        for all $v \in \mathcal{N}(u)$ in parallel do
23:            if $status_{u,v} \neq$ "meeting points" AND $alreadyRewound_{u,v} = 0$ then
24:                if $|T_{u,v}| > minChunk$ then
25:                    Send a rewind message to $v$ and truncate $T_{u,v}$ by one chunk
26:                    $alreadyRewound_{u,v} \leftarrow 1$
27:            if a rewind message is received from $v$ then
28:                if $status_{u,v} \neq$ "meeting points" AND $alreadyRewound_{u,v} = 0$ then
29:                    Truncate $T_{u,v}$ by one chunk
30:                    $alreadyRewound_{u,v} \leftarrow 1$
```

Algorithm 6 InitializeState()   (for party u, simulation against oblivious noise without a CRS)

```
1: $K \leftarrow m$
2: for all neighbors $v \in \mathcal{N}(u)$ in parallel do
3:     Initialize $T_{u,v} = \emptyset$
4:     Initialize $k_{u,v}, E_{u,v}, mpc1_{u,v}, mpc2_{u,v} \leftarrow 0$
5:     $status_{u,v} \leftarrow$ "simulate"
6:     $alreadyRewound_{u,v} \leftarrow 0$
7:     $\delta := 2^{-\Theta(|\Pi|K/m)}$
8:     $S := (S_{i,u,v})_{i\in[100|\Pi|]} \leftarrow$ RandomnessExchange($u$, $v$, $\delta$, $K$)
9: $status_u \leftarrow 1.$
10: $netCorrect_u \leftarrow 1$
```

Algorithm 7 RandomnessExchange(u, v, δ, K)

```
    input: $u$ calling party, $v \in \mathcal{N}(u)$. $\delta > 0$ is a parameter that specifies the bias we desire from our random string.

1: Fix an arbitrary total order on the vertices in $V$ (common for all instantiations of this procedure).
2: $\ell \leftarrow \Theta(|\Pi|K) \cdot 100|\Pi|$,
3: $r \leftarrow \Theta(\log(1/\delta) + \log(\ell))$, large enough to generate a string of length at least $\ell$ and bias at most $\delta$ via Lemma 2.3.5.
4: $C \leftarrow$ error correcting code with block length $\Theta(r)$, constant rate and constant distance (e.g., the code from Theorem 2.3.3)
5: if $u < v$ in the ordering of $V$ then
6:     Sample a uniformly random string $L \in \{0,1\}^r$
7:     Send $C(L)$ to $v$
8: else
9:     Receive $W$ from $v$
10:    $L \leftarrow C^{-1}(W)$, the decoding of $W$ with respect to the code $C$.
11: $S_{u,v} \leftarrow \left(\{0,1\}^{\Theta(|\Pi|K)}\right)^{100|\Pi|}$ gets the $\delta$-biased string of length $\ell$ generated from $L$ by Lemma 2.3.5.
12: Output $S_{u,v}$.
```

Algorithm 8 FlagPassing(u, $status$) (from Section 2.2)

Let $\rho \in V$ be a specific node known by all the parties. Let $\mathcal{T}$ be a spanning tree generated by a breadth-first-search starting from ρ. Denote the depth of $\mathcal{T}$ as $d(\mathcal{T})$, where the depth of a single vertex is 1. Finally, let the *level* of a vertex be defined as $\ell(v) := \ell(u) + 1$, where u is the parent of v in $\mathcal{T}$, and $\ell(\rho) = 1$.

```
1: $netCorrect \leftarrow status$
2: **if** $u$ is a leaf in $\mathcal{T}$ **then**
3:     Send $netCorrect$ to parent vertex in $\mathcal{T}$.
4:     Sleep for $\ell(u) - 1$ rounds.
5: **else**
6:     Sleep for $d(\mathcal{T}) - \ell(u)$ rounds. Ignore any messages received in these rounds.
7:     Receive $b_1, \dots, b_k$, one symbol from each child in $\mathcal{T}$.
8:     $netCorrect \leftarrow \bigwedge_{i=1}^{k} b_i \wedge status$
9:     **if** $u \neq \rho$ **then**
10:        Send $netCorrect$ to parent.
11:        Sleep for $\ell(u) - 1$ rounds. Ignore any messages received in these rounds.
12: **if** $u = \rho$ **then**
13:     Send $netCorrect$ to children.
14: **else**
15:     Sleep for $\ell(u) - 1$ rounds. Ignore any messages received in these rounds.
16:     Receive $b$ from parent.
17:     $netCorrect \leftarrow b \wedge status$
18:     **if** $u$ is not a leaf in $\mathcal{T}$ **then**
19:         Send $netCorrect$ to children.
20:         Sleep for $d(\mathcal{T}) - \ell(u)$ rounds. Ignore any messages received in these rounds.
21: **return** $netCorrect$
```

Algorithm A: Analysis

We now analyze Algorithm A and show that for any oblivious adversary that corrupts at most $O(\varepsilon/m)$ fraction of the communication, the parties correctly simulate Π with high probability.

The main theorem of this section is as follows. Its proof will also be useful in Section 2.3.4.

Theorem 2.3.9. *Assume a network $G = (V, E)$ with $n = |V|$ parties and $m = |E|$ links. Suppose Π is a multiparty protocol over the network G with communication complexity $\mathsf{CC}(\Pi)$, binary alphabet and fixed order of speaking. Let $|\Pi| = \frac{\mathsf{CC}(\Pi)}{5m}$ and let $\varepsilon > 0$ be a sufficiently small constant. Consider an instance of Algorithm A with an oblivious adversary and let CC denote the communication complexity of the instance. Then, the probability that the total number of errors satisfies $\mathsf{Err} \le \frac{\varepsilon}{m} \cdot \mathsf{CC}$ and yet Algorithm A either simulates Π incorrectly or has $\mathsf{CC} = \omega(\mathsf{CC}(\Pi))$, is at most $\exp(-\Omega(|\Pi|))$.*

Overview The analysis we perform resembles the one of Algorithm 1, that we analyze in Section 2.2. We start by providing an overview of the parts of our analysis from Section 2.2 that remain exactly the same; for full details, see Section 2.2.4.

We recall Eq. (2.6), which states the potential function used to track the progress of the parties in Section 2.2:

$$\phi \overset{\text{def}}{=} \sum_{(u,v)\in E} \left(\frac{K}{m} G_{u,v} - K \cdot \varphi_{u,v}\right) - C_1 K B^* + C_7 K \cdot \mathsf{EHC}, \tag{2.10}$$

where we refer the reader to Fig. 2-1 for details on the meaning of the terms. Further, we note that we set $K = m \log(m)$ in this section, as opposed to $K = m$ in Section 2.2.

Analogous to Section 2.2, Theorem 2.3.9 effectively follows from two lemmas about the potential function ϕ. The first lemma says that the value of ϕ increases by at least $\Omega(K)$ in every iteration. This lemma was written for Algorithm 1 in Section 2.2, but does not rely on the implementation of InitializeState(), so it applies to Algorithm A with an identical proof.

Lemma 2.3.10 (Lemma 2.2.3 for Algorithm A)**.** *Fix any iteration of Algorithm A and let b be the number of links with errors or hash collisions in this iteration. Then the following holds.*

1. *The potential ϕ increases by at least K in this iteration.*

2. *The amount of communication in the entire network during this iteration (*CC*) satisfies*

$$\mathsf{CC} \le \alpha(1+b)K$$

   where α is a sufficiently large constant.

The second lemma we need bounds the parameter EHC, that is, bounds the total amount of errors and hash collisions, assuming that the adversary follows its budget. This lemma is required in order to show that when the potential function is sufficiently large, the protocol has been simulated correctly (i.e., the high potential doesn't stem from high EHC). Unfortunately, we cannot bound the number of hash collisions the same way we did in Section 2.2. The reason is that we used uniformly random seeds in Algorithm 1 from Section 2.2, whereas in Algorithm A they come from a δ-biased distribution.

We develop a new bound on the hash collisions in this setting by relating the number of hash collisions that occur when the parties share δ-biased seeds to the case where the parties share uniformly random seeds. More specifically, we consider an arbitrary fixed pattern of hash agreements and misses throughout the entire protocol (and throughout the entire network). Fixing the pattern of hash agreements and misses additionally fixes the transcripts throughout the entire protocol, as this is the only randomness in the protocol. This allows us to apply Lemma 2.3.6 to relate the probability of this pattern of hash agreements and misses when the parties share uniformly random seeds to the corresponding probability of the same pattern with δ-biased seeds.

Finally, we will need to establish that the adversary cannot meaningfully corrupt the randomness exchange protocol. That is, the adversary cannot make parties use mis-matching δ-biased strings. Since the adversary has a limited budget of corrupting ε/m-fraction of the communication, the randomness exchange succeeds if the budget does not suffice to corrupt a single coded message (Line 4 in Algorithm 7).

To this end we need to bound the communication of Algorithm A, and thus, the budget of the adversary. Note that the communication depends on the randomness exchange, that is, it is possible that corrupting this part and making parties use mis-matching δ-biased strings will lead to a large amount of communication, which will allow this corruption. Hence, we need to bound the communication of Algorithm A *assuming that some of the δ-biased strings are adversarially chosen*.

We show that despite the fact that the adversary controls some of these seeds, the communication does not increase too much, and the increase is proportional to the number of seeds controlled by the adversary. This bound on the communication effectively bounds the budget of the adversary. By choosing the right parameters, we establish that a noise level of ε/m-fraction of the communication is insufficient to corrupt the δ-biased string of even a single link.

We put the results above together to prove the desired bound on the number of hash collisions in Corollary 2.3.31. We use this to prove Theorem 2.3.9 at the end of this section, which follows with a proof identical to that of Theorem 2.2.2 in Section 2.2.6.

Bounding hash collisions: Goal and Relevant Notations As mentioned in the overview, the adversary has two types of errors it can place: it can use errors to tamper with the randomness exchange phase (and make two adjacent parties fail to share a random seed, by making so many errors that the seed is incorrectly decoded), and it can use errors to corrupt the "main" part of Algorithm A (i.e. after InitializeState() is completed). Let E' be the set of edges on which the parties successfully share δ-biased randomness after InitializeState() is completed; intuitively, only edges in E' can simulate correctly, since edges in $E \setminus E'$ will be comparing hashes computed with different seeds, and will get nonsensical results. We let CC and Err denote the *total* amount of communication and number of errors respectively in the entire protocol, and we define CC$'$ and Err$'$ be the amount of communication and number of errors during the main part of Algorithm A.

Our main goal in this section is to bound the number of hash collisions that occur even when the parties use δ-biased randomness. Towards this end, we recall the following proposition about $\varphi_{u,v}$ from Section 2.4, slightly rephrased.

Proposition 2.3.11 (Proposition 2.4.2). *$\varphi_{u,v}(i)$ is nonnegative, and equals 0 only if $T_{u,v}(i) = T_{v,u}(i)$.*

In other words, if u and v are consistent at the beginning of iteration i, then $\varphi_{u,v} = 0$.

We also re-define a dangerous triple (recall Definition 2.2.12) in a natural way, only considering the edges E' on which the parties have shared the random seed correctly.

Definition 2.3.12 (Dangerous Triple). *Given an iteration $i \in [100|\Pi|]$ and an edge $(u,v) \in E$, we say that the triple (i,u,v) is* dangerous *if $B_{u,v}(i) > 0$ and $(u,v) \in E'$.*

A dangerous triple denotes an iteration i and edge $(u,v) \in E'$ in which the transcripts of u and v differ. This is natural, as edges in $E \setminus E'$ are ones in which the parties did not correctly share a seed for hashing, and so we have no control over what happens on these links. Our goal is to bound the number of dangerous triples, as this bounds the number of possible hash-collisions (recall, that a hash collision can happen only when the partial transcripts $T_{u,v}$ and $T_{v,u}$ differ).

Note that, due to Proposition 2.4.2, we have that the number of dangerous triples is upper bounded by the number of triples (i,u,v) in which $\varphi_{u,v}(i)$ is positive for some link $(u,v) \in E'$. We will let D denote the total number of triples for which $\varphi_{u,v}(i) > 0$ throughout the entire execution of Algorithm A. It suffices to upper bound D.

In Section 2.2, we proved that if the parties share a (long) *uniform* random string, then the probability that D is much larger than Err' is extremely small. However, in this part of the work, the parties instead share a (short) δ-biased random string. We need to show that this has no effect on the correctness of the coding scheme.

Below, we state our main lemma of this section, which proves exactly the above. Namely, we consider the (bad) event that D is much larger than Err', and bound its probability in the δ-biased setting via its probability the uniform randomness setting.

Lemma 2.3.13. *Consider an instance of Algorithm A with $|\Pi|K \geq m\log(m)$, and let p denote the collision probability of the inner product hash function used by the protocol under uniformly random seeds. Suppose the adversary is oblivious, and let k be a real number. Then*

$$\mathbb{P}_S[D > k\mathsf{Err}'] \leq e \cdot p^{-2\mathsf{Err}} \cdot \mathbb{P}_R[D > k\mathsf{Err}']$$

where $S = (S_{i,u,v}, S_{i,v,u})_{i\in[100|\Pi|],(u,v)\in E}$ is the strings shared by each party after Algorithm 7, and where $R = (R_{i,u,v}, R_{i,v,u})_{i\in[100|\Pi|],(u,v)\in E}$ is sampled from a distribution such that for every $(u,v) \in E'$ the parties share a uniform string, and for any $(u,v) \notin E'$ the seeds are sampled identically to the distribution of S on the corresponding links.

In order to use the lemma above in a meaningful way, we need to limit the number of errors Err' that the adversary can commit in the main part of the protocol. We show below that the ratio of errors to communication within the *main part* of the protocol is always within a constant factor of the ratio of errors to communication in the protocol *overall*. In order to do this, we use Claim 2.3.27 to bound the communication in the randomness exchange phase; the formal statement of this claim and its simple proof can be found later in this subsection.

Claim 2.3.14. *Let CC be the communication complexity of an execution of Algorithm A and let $\varepsilon > 0$ be a sufficiently small constant. Assume that $|\Pi|K \geq m\log(m)$. If $\mathsf{Err} \leq \frac{\varepsilon}{K} \cdot \mathsf{CC}$, then $\mathsf{Err} \leq \frac{\varepsilon'}{K} \cdot \mathsf{CC}'$, where $\varepsilon' = \Theta(\varepsilon)$.*

Proof. Claim 2.3.27 shows that the communication in the randomness exchange is $\Theta(|\Pi|K)$ as long as $|\Pi|K \geq m\log(m)$. Furthermore, the communication in the main part of the protocol is always at least $\Omega(|\Pi|K)$: there are $100|\Pi|$ iterations, and the communication in each meeting-points phase is at least $\Theta(K)$, since the parties pass hashes of size $\Theta(K/m)$ to each other and there are m links. □

Due to Claim 2.3.14, we can consider the main part of the protocol in isolation when bounding the number of dangerous triples, despite the fact that there is extra communication in the randomness exchange phase.

We now make a definition that is critical to our proof of Lemma 2.3.13.

Definition 2.3.15 (Agreement patterns)**.** *An agreement pattern $a \in \{0,1\}^{100|\Pi|\cdot(2|E|)}$ consists of a pair of elements $(a_{i,u,v}, a_{i,v,u})$ for each iteration $i \in [100|\Pi|]$ and each edge $(u,v) \in E$, where $a_{i,u,v} = 1$ if and only if party u* ***believes*** *that u and v have matching hashes in iteration i. The term $a_{i,v,u}$ is defined analogously.*

Note that $a_{i,u,v} = 1$ does not imply that the hashes *actually* match. Indeed, in the presence of channel noise the parties may believe that their hashes match (or mismatch) while the real hashes do not.

To be precise, multiple hashes are passed by u and v in iteration i, and an agreement pattern should account for misses/matches in all of them. To ease the readability of the analysis, we will assume that the parties exchange only *a single* hash, and that they make all their decisions according to whether this hash missed or matched. This will not affect the validity of the analysis, yet it will somewhat simplify it. We elaborate on this simplifying assumption in Remark 2.3.18 in the proof of Lemma 2.3.13.

Given an agreement pattern a, let $a_{u,v} \in \{0,1\}^{200|\Pi|}$ denote the vector

$$a_{u,v} \overset{\text{def}}{=} ((a_{1,u,v}, a_{1,v,u}), \ldots, (a_{100|\Pi|,u,v}, a_{100|\Pi|,v,u})),$$

which is the pattern of hash misses and matches that both u and v see along a single link (u,v) throughout the protocol. Similarly, let

$$a_i \overset{\text{def}}{=} (a_{i,u,v}, a_{i,v,u})_{(u,v)\in E} \in \{0,1\}^{2|E|}$$

denote the vector of hash misses and matches for every party and every link in an iteration i, and let $a_{\leq i} \overset{\text{def}}{=} (a_1, \ldots, a_i)$. Call $a_i, a_{u,v}$, and $a_{\leq i}$ *sub-patterns* of a.

Note that, since we fix the oblivious adversary's errors ahead of time, the agreement pattern a observed in a given instance of Algorithm A is a function of the random seed S that the algorithm samples. Let A denote the random variable over agreement patterns a, where $\mathbb{P}_S[A = a]$ is the probability, over the random string S, that the protocol gets agreement pattern a, where S is the concatenation of the randomness used by each party for each link. Similar to previous notation, let A_i denote the random variable A restricted to sub-patterns a_i, and $A_{\leq i}$ denote the restriction to sub-patterns $a_{\leq i}$.

Bounding hash collisions: Detailed proof In this section, we prove Lemma 2.3.13. First, we state a simple but vital claim regarding agreement (sub-)patterns.

Claim 2.3.16. *Fix the inputs to the protocol and an oblivious adversary, let i be an iteration of Algorithm A, and fix an agreement sub-pattern $a_{\leq i}$ for the protocol. Then for all $(u,v) \in E$ and all $j = 1,\ldots,i+1$, the partial transcripts $T_{u,v}(j)$ and $T_{v,u}(j)$ are fixed.*

Proof. We will prove this by induction on j. In fact, we prove something stronger - for every party u and every variable var that u keeps in its state, $var(j)$ is fixed.

For $j = 1$, it is clear the statement holds, as no communication has happened yet.

Now assume that the claim holds for $j < i+1$. In iteration j, the parties exchange hashes of the partial transcripts $T_{u,v}(j)$ and $T_{v,u}(j)$. Since $j \leq i$, we have fixed a_j, which means we have already fixed whether or not u and/or v observe hash collisions for *every* pair of adjacent parties $(u,v) \in E$. Noting that the *only randomness in Algorithm A comes from hashing*, we can conclude that this fixes *every action of each party* in iteration j.

Therefore, the values of every state variable (including the partial transcripts $T_{u,v}$ and $T_{v,u}$) at the beginning of iteration $j+1$ are fixed. □

The following corollary follows by taking $i = 100|\Pi|$.

Corollary 2.3.17. *Fix the inputs to the protocol and an oblivious adversary, and fix an agreement pattern a. For all $(u,v) \in E$ and all iterations i, $T_{u,v}(i)$ and $T_{v,u}(i)$ are fixed.*

Remark 2.3.18 (Remark on Hash Agreements)**.** *The situation is actually somewhat more complex than just whether or not the hashes of the transcript $T_{u,v}(i)$ matches the (possibly corrupted) hash of $T_{v,u}(i)$ in iteration i. In the meeting-points protocol, the parties send each other* two *hashes of transcripts, one for each of their current "meeting points." Party u compares these two received hashes to her own two hashes, and takes different outcomes depending on* *how these received hashes match with her hashes. The parties also send each other hashes of their meeting-point iteration k, and compare these. Hence, the actions of party u are actually a function of these 5 hash comparisons, and the agreement pattern should be defined over 2^5-ary symbols rather than bits. For simplicity, we pretend as if the action is a function of just one hash comparison for this proof. This only affects the constant in the exponent of p in Lemma 2.3.13.*

Now we can prove Lemma 2.3.13.

Proof of Lemma 2.3.13. Fix the inputs to the parties and the adversary's errors. By Corollary 2.3.17, fixing the agreement pattern throughout the entire protocol fixes all the partial transcripts $T_{u,v}(i), T_{v,u}(i)$ for each link $(u,v) \in E$ and iteration i. In other words, as long as the agreement pattern for two runs of the Algorithm A are the same, the actions of the parties in each phase of the two instances are identical as well, *even if* the random seeds sampled in the two runs are different. In other words, every random seed that leads to agreement pattern a *also* leads to the exact same set of pairs of transcripts $\{(T_{u,v}(i), T_{v,u}(i))\}_{i,u,v}$ for every iteration i and edge $(u,v) \in E$.

We briefly note that it is possible that an agreement pattern a never occurs in Algorithm A (i.e., $\mathbb{P}[A = a] = 0$ for any distribution of randomness). As a simple example, consider the cases where no noise occurs at all, and yet $a_{i,u,v} = 0$ for some i and $(u,v) \in E$. This is obviously impossible since the absence of noise implies that the transcripts at both sides are the same, and hence their hashes must be the same. Since no noise is present, both parties receive the correct hash values and the agreement must be $a_{i,u,v} = 1$ in all iterations.

We call an agreement pattern a *consistent* with respect to a distribution $\mathcal{D}$ if there is some random seed $S \in \text{supp}(\mathcal{D})$ that leads to the agreement pattern a. If a is consistent with the uniform distribution, then we simply say a is consistent. Given a consistent agreement pattern a, denote the unique set of pairs of transcripts that can coexist with it in the protocol as $\{(T^{(a)}_{u,v}(i), T^{(a)}_{v,u}(i))\}_{i,u,v}$. We have already argued that a can coexist with at most one set of pairs of transcripts, since fixing a fixes the behavior of all the parties in the protocol.

Now we prove the key proposition for this proof, which roughly says that the probability that we see a consistent agreement pattern a in Algorithm A is equal to the probability of the following event:

$$h_u(T^{(a)}_{u,v}) = h_v(T^{(a)}_{v,u}) \text{ if and only if } a_{i,u,v} = 1 \text{ for all } (i,u,v),$$

where h_u denotes the hash function that u uses during i with seed $S_{i,u,v}$, h_v denotes the hash function that v uses with seed $S_{i,v,u}$, and $\{(T^{(a)}_{u,v}(i), T^{(a)}_{v,u}(i)\}$ denotes the unique set of pairs of transcripts that is consistent with a given the specific oblivious noise.

This alone is not quite accurate, since u's actions are not determined by whether the hashes of $T^{(a)}_{u,v}$ and $T^{(a)}_{v,u}$ *actually* match, but rather by whether u *thinks* they matched. These two events can be different when the adversary corrupts the hash that v sends to u, which is why we defined $a_{i,u,v}$ and $a_{i,v,u}$ separately. Hence, instead of comparing $a_{i,u,v}$ to the event that $h(T^{(a)}_{u,v}(i)) = h(T^{(a)}_{v,u}(i))$, we actually compare $a_{i,u,v}$ to the event that $h(T^{(a)}_{u,v}(i)) = \widetilde{h}(T^{(a)}_{v,u}(i))$, where $\widetilde{h}(\cdot)$ is the function that first applies the hash h to the input and then induces errors from the adversary.

For notational convenience, let $Z(S,i,u,v,a)$ denote the event that the hash agreement between u and v in iteration i is consistent with a, when seeded by S. Namely, $Z(S,i,u,v,a)$ is the following event

$$Z(S,i,u,v,a) \overset{\text{def}}{=} \left\{ \left(1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i)) = \widetilde{h}_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,u,v} \right) \bigwedge \left(1_{\widetilde{h}_{S,i,u,v}(T^{(a)}_{u,v}(i)) = h_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,v,u} \right) \right\} \tag{2.11}$$

where $h_{S,i,u,v}$ denotes the hash function h used in Algorithm A when given the seed $S_{i,u,v}$, and $\widetilde{h}_{S,i,u,v}(\cdot)$ is defined by applying $h_{S,i,u,v}$ to its input, then modifying the output with the errors that the adversary commits on the transmission from u to v in iteration i.

Proposition 2.3.19. *Fix a distribution $\mathcal{D}$ and a consistent agreement pattern a (with respect to the uniform distribution). Fix $\{(T^{(a)}_{u,v}(i), T^{(a)}_{v,u}(i))\}_{i,u,v}$ to the unique transcripts forced in the protocol by fixing a. Let $S = (S_{i,u,v}, S_{i,v,u})_{i\in[100|\Pi|,(u,v)\in E]}$ denote the tuple of random strings used by the main part of the protocol.*

Then,

$$\mathbb{P}_{S\sim\mathcal{D}}[A = a] = \mathbb{P}_{S\sim\mathcal{D}}\left[\bigwedge_{i,u,v} Z(S,i,u,v,a)\right], \tag{2.12}$$

Furthermore, if a is not consistent with $\mathcal{D}$, then Eq. (2.12) *is equal to 0.*

Proof. We show that the set of randomly sampled strings S that gives rise to the agreement pattern a in the protocol is the same set of strings that gives rise to $1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=\widetilde{h}_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,u,v}$ and $1_{\widetilde{h}_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,v,u}$ for all (i,u,v). Therefore, the probability on both sides of Equation 2.12 is simply the measure of S under the distribution $\mathcal{D}$, and the result follows as a corollary.

We proceed by induction on i, the iteration of the protocol. For $i = 1$, note that all the pairs transcripts match at the beginning of the iteration, as no communication has occurred yet. Hence, the transcripts at the beginning of the protocol between u and v are trivially exactly $T^{(a)}_{u,v}(1)$ and $T^{(a)}_{v,u}(1)$. Hence, $A_{\leq 1} = a_{\leq 1}$ if and only if the strings $S_{1,u,v}, S_{1,v,u}$ used to hash the transcripts in the first iteration satisfy that $1_{h_{S,1,u,v}(T^{(a)}_{u,v}(1))=\widetilde{h}_{S,1,v,u}(T^{(a)}_{v,u}(1))} = a_{1,u,v}$ and $1_{\widetilde{h}_{S,1,u,v}(T^{(a)}_{u,v}(1))=h_{S,1,v,u}(T^{(a)}_{v,u}(1))} = a_{1,v,u}$ for all $(u,v) \in E$. I.e.,

$$\mathbb{P}_{S\sim\mathcal{D}}[A_{\leq 1} = a_{\leq 1}] = \mathbb{P}_{S\sim\mathcal{D}}\left[\bigwedge_{(u,v)\in E} Z(S,1,u,v,a)\right].$$

Now suppose inductively that the set of random strings $\mathcal{S}_{i-1}$ that lead to $A_{\leq i-1} = a_{\leq i-1}$ is the same as the set of strings that lead to

$$1_{h_{S,j,u,v}(T^{(a)}_{u,v}(j))=\widetilde{h}_{S,j,v,u}(T^{(a)}_{v,u}(j))} = a_{j,u,v}$$

and

$$1_{\widetilde{h}_{S,j,u,v}(T^{(a)}_{u,v}(j))=h_{S,j,v,u}(T^{(a)}_{v,u}(j))} = a_{j,v,u}$$

for all (j,u,v) s.t. $j \leq i-1$. Note that, since all $S \in \mathcal{S}_{i-1}$ lead to the same agreement sub-pattern $a_{\leq i-1}$, they also all lead the protocol to the pairs of transcripts $(T^{(a)}_{u,v}(i), T^{(a)}_{v,u}(i))$ at iteration i, since fixing the $a_{\leq i-1}$ fixes the behavior of all the parties up until the

ith iteration of the protocol. Hence, the random strings that lead to $A_{\leq i} = a_{\leq i}$ are precisely the strings in $\mathcal{S}_{i-1}$ that additionally lead to $1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=\widetilde{h}_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,u,v}$ and $1_{\widetilde{h}_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,v,u}$ for all (u,v).

Applying the inductive hypothesis, $\mathcal{S}_{i-1}$ is exactly the set of strings that lead to $1_{h_{S,j,u,v}(T^{(a)}_{u,v}(j))=\widetilde{h}_{S,j,v,u}(T^{(a)}_{v,u}(j))} = a_{j,u,v}$ and $1_{\widetilde{h}_{S,j,u,v}(T^{(a)}_{u,v}(j))=h_{S,j,v,u}(T^{(a)}_{v,u}(j))} = a_{j,v,u}$ for all $(u,v) \in E$ and all $j \leq i-1$. Hence we conclude that the the set of strings that lead to $A_{\leq i} = a_{\leq i}$ is the same as the set of strings that lead to $1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=\widetilde{h}_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,u,v}$ and $1_{\widetilde{h}_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,v,u}(T^{(a)}_{v,u}(i))} = a_{i,v,u}$ for all (u,v) and $j \leq i$.

Finally, the last claim of the Proposition follows trivially, since if a is not consistent with $\mathcal{D}$, then no $S \in \text{supp}(\mathcal{D})$ leads to a, and we have $\mathbb{P}_{S\sim\mathcal{D}}[A = a] = 0$ by definition. □

Now we can bound the probability of getting too many dangerous triples with δ-biased randomness from the randomness exchange (Algorithm 7) in terms of the probability of getting too many dangerous triples with uniform randomness. Recall that, after fixing a consistent agreement pattern a, there is exactly one possible setting to all pairs of transcripts $(T_{u,v}(i), T_{v,u}(i))$ for each link and iteration that complies with this agreement pattern. Given such a setting of pairs of transcripts, it is easy to read off an upper bound on the number of dangerous triples, since in every dangerous triple (i,u,v), we have that $T_{u,v}(i) \neq T_{v,u}(i)$ and $(u,v) \in E'$. Recall that we are interested in the probability there where more than $k\mathsf{Err}$ dangerous triples, where k is a real number given by the lemma's statement.

Definition 2.3.20. *A consistent agreement pattern a is called* bad *if it leads to having more than $k\mathsf{Err}$ dangerous triples in the considered instance.*

We let $\mathsf{Bad} = \{a \text{ is bad and consistent}\}$ be the set of all bad consistent agreement patterns (given the fixed inputs and fixed error pattern). The, for any distribution D we can write

$$\mathbb{P}_{S\sim\mathcal{D}}[D > k\mathsf{Err}] = \sum_{a\in\mathsf{Bad}} \mathbb{P}_{S\sim\mathcal{D}}[A = a]. \tag{2.13}$$

Hence, to prove Lemma 2.3.13, it suffices to show that

$$\sum_{a\in\mathsf{Bad}} \mathbb{P}_{S\sim\mathcal{D}_S}[A=a] \le e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}} \mathbb{P}_{R\sim\mathcal{D}_R}[A=a] \tag{2.14}$$

where we recall that $\mathcal{D}_S$ is the distribution where the $\{S_{u,v}\}$ are drawn i.i.d from a δ-biased distribution for $(u,v)\in E'$ while for $(u,v)\in E\setminus E'$ the string $\{S_{u,v}, S_{v,u}\}$ are adversarially chosen by the fixed noise. The distribution $\mathcal{D}_R$ is the same for $(u,v)\in E\setminus E'$ and uniform for $(u,v)\in E'$, as described in the statement of the lemma.

Since $\mathbb{P}_{R\sim\mathcal{D}_R}[A=a]=0$ for any agreement pattern a that is not consistent w.r.t $\mathcal{D}_R$, we can sum the right-hand side Eq. (2.14) only over a's that are bad and consistent w.r.t $\mathcal{D}_R$. Denote these by the set

$$\mathsf{Bad}_{\mathcal{D}_R} = \{a \mid a\in \mathsf{Bad} \text{ and is consistent with respect to } \mathcal{D}_R\}.$$

Noting that the uniform distribution trivially contains the support of $\mathcal{D}_R$, which itself contains the support of $\mathcal{D}_S$, we get from Eq. (2.14) that

$$\sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{S\sim\mathcal{D}_S}[A=a] \le e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{R\sim\mathcal{D}_R}[A=a] \tag{2.15}$$

For ease of notation in the proof below, let $Q(a)$ denote the probability of seeing the agreement sub-pattern $a_{u,v}$ for all $(u,v)\in E\setminus E'$. Formally:

$$Q(a) \stackrel{\text{def}}{=} \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i,(u,v)\in E\setminus E'} Z(S,i,u,v,a)\right] \tag{2.16}$$

where the events $Z(S,i,u,v,a)$ are as defined in Eq. (2.11).

Now we proceed to use Proposition 2.3.19 to relate the probability of seeing a bad, consistent agreement pattern a to the probability of seeing the same agreement pattern when hashing a fixed input, which will let us apply Lemma 2.3.6. We upper bound the left-hand side of Eq. (2.15) as follows:

$$
\begin{aligned}
&\sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{S\sim\mathcal{D}_S}[A=a] \\
&= \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i,(u,v)\in E} Z(S,i,u,v,a)\right] \\
&= \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \left(\prod_{(u,v)\in E'} \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i} Z(S,i,u,v,a)\right]\right)\cdot Q(a) \\
&\le \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \left(\prod_{(u,v)\in E'} \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i:\text{no err on }(i,u,v)} Z(S,i,u,v,a)\right]\right)\cdot Q(a) \\
&= \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \left(\prod_{(u,v)\in E'} \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i:\text{no err on }(i,u,v)} 1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v}\right]\right)\cdot Q(a) \qquad (2.17)
\end{aligned}
$$

The first equality is Proposition 2.3.19. The second equality is due to the fact that, given a specific a and fixing the adversary's errors, the event $\bigwedge_i Z(S,i,u,v,a)$ for $(u,v)\in E'$ is independent from $\bigwedge_i Z(S,i,u',v',a)$ for all $(u',v')\in E$ such that $(u',v')\neq(u,v)$, since $S_{u,v}$ is sampled independently from $S_{u',v'}$. To elaborate, once a is fixed, all the transcripts $\{T^{(a)}_{u,v}(i)\}$ are fixed as well (Claim 2.3.16), and since the channel's noise is fixed we get that, for any (i',u',v'), the event $Z(S,i',u',v',a)$ depends only on the seed $S_{u',v'}$.

The inequality in the third line follows from the fact that the event $\bigwedge_{i:\text{no err on }(i,u,v)} Z(S,i,u,v,a)$ contains the event $\bigwedge_i Z(S,i,u,v,a)$, since we perform AND over fewer random variables. Note that, by i : no err on (i,u,v), we mean that there is no adversarial error between u and v in the entire iteration i.

The last equality follows from the following facts: (1) the parties successfully share a seed on the edges in E', so $S_{i,u,v}=S_{i,v,u}$ for all $(u,v)\in E'$, (2) $h_{S,i,u,v}=\widetilde{h}_{S,i,u,v}$ and there is no adversarial error between u and v in iteration i, and (3) $a_{i,u,v}=a_{i,v,u}$ when a is consistent and there is no error on (i,u,v), because the hashes are transmitted properly, and so either u and v will both have a hash match or both have a hash miss, and so if $a_{i,u,v}\neq a_{i,v,u}$ then a would not be consistent.

Now we can apply Lemma 2.3.6 to the right-hand side of Eq. (2.17). Note that

$$\bigwedge_{i:\text{no err on }(i,u,v)} 1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v} \tag{2.18}$$

is an event about a sequence of hash misses and matches on a fixed input when using a seed that is chopped up from a string $S_{u,v}$ sampled from a δ-biased source. By Lemma 2.3.6, the probability of this event cannot differ by much from the same event given the seed is uniform,

$$\left| \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i:\text{no err on }(i,u,v)} 1_{h_{S,i,u,v}(T^{(a)}_{u,v}(i))=h_{S,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v}\right] \right.$$
$$\left. -\mathbb{P}_{R\sim\mathcal{D}_R}\left[\bigwedge_{i:\text{no err on }(i,u,v)} 1_{h_{R,i,u,v}(T^{(a)}_{u,v}(i))=h_{R,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v}\right]\right| \le \delta \tag{2.19}$$

Therefore, we bound the right-hand side of Eq. (2.17) by

$$\le \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \left(\prod_{(u,v)\in E'}\left(\mathbb{P}_{R\sim\mathcal{D}_R}\left[\bigwedge_{i:\text{no err on }(i,u,v)} 1_{h_{R,i,u,v}(T^{(a)}_{u,v}(i))=h_{R,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v}\right]+\delta\right)\right)\cdot Q(a)$$
$$= \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \left(\prod_{(u,v)\in E'}\left(\mathbb{P}_{R\sim\mathcal{U}}\left[\bigwedge_{i:\text{no err on }(i,u,v)} Z(R,i,u,v,a)\right]+\delta\right)\right)\cdot Q(a) \tag{2.20}$$

where we use the definition of $Z(R,i,u,v,a)$ to avoid cumbersome notation. Note that in the last transition we replaced the distribution $\mathcal{D}_R$ with a uniform distribution $\mathcal{U}$. This follows since, by the definition of $\mathcal{D}_R$, each of the seeds $R_{i,u,v}$ with $(u,v)\in E'$ is uniform and independent.

We now wish to bound δ as a fraction of the probability of the event (Eq. (2.18)).

Claim 2.3.21.

$$\delta \le 2^{-|\Pi|K/m} \min_{a_{u,v}:(u,v)\in E'} \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\textit{no err on }(i,u,v)} Z(R,i,u,v,a)\right]. \tag{2.21}$$

Proof. Recall that, given a uniform seed, the hash collision probability is exactly $p = 2^{-\Theta(K/m)}$ (Lemma 2.1.3). Note that in the event of Eq. (2.18) we are AND-ing over at most $\Theta(|\Pi|)$ iterations i. Then we can write,

$$\begin{aligned}\min_{a,(u,v)\in E'} \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{no err on }(i,u,v)} Z(R,i,u,v,a)\right] &\geq \min_{a,(u,v)\in E'} \prod_{i:\text{no err on }(i,u,v)} \mathbb{P}_{R_{u,v}\sim\mathcal{U}}[Z(R,i,u,v,a)] \\ &\geq p^{|\Pi|} \\ &= 2^{-\Theta(|\Pi|\cdot K/m)}.\end{aligned}$$

Eq. (2.21) holds by recalling that $\delta = 2^{-\Theta(|\Pi|\cdot K/m)}$ and taking the constant in the exponent of δ to be sufficiently large. □

Plugging Eq. (2.21) into Eq. (2.20), the latter can be bounded by

$$\begin{aligned}&\leq \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\left(1+2^{-|\Pi|K/m}\right)\cdot \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{no err in }(u,v))} Z(R,i,u,v,a)\right]\right)\cdot Q(a) \\ &\leq \left(1+2^{-|\Pi|K/m}\right)^m \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{ no err on }(i,u,v)} Z(R,i,u,v,a)\right]\right)\cdot Q(a) \\ &\leq e\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{ no err on }(i,u,v)} Z(R,i,u,v,a)\right]\right)\cdot Q(a) \qquad (2.22)\end{aligned}$$

where in the last transition, we use the fact that $|\Pi|K \geq m\log m$ to conclude that $(1 + 2^{-|\Pi|K/m})^m \leq (1+\frac{1}{m})^m \approx e$.

We are almost done: if Eq. (2.22) had an AND over *all* iterations i instead of just the iterations with no error, we would be able to apply Proposition 2.3.19 on this term and bound it via which fits the right-hand side of the claim we are proving (Eq. (2.17)). Note that

$$\sum_{a \in \mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_R[A = a]$$
$$= \sum_{a \in \mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_R\left[\bigwedge_{i,(u,v)\in E} Z(R,i,u,v,a)\right]$$
$$= \sum_{a \in \mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{no err on }(i,u,v)} Z(R,i,u,v,a)\right] \times \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{err on }(i,u,v)} Z(R,i,u,v,a)\right] \cdot Q(a)$$

where we use Proposition 2.3.19 in the first transition and the fact that the $R_{i,u,v}$ are independently sampled uniform strings when $(u,v) \in E'$ in the second.

The extra term can be lower bounded as follows.

Claim 2.3.22. *For any $a \in \mathsf{Bad}_{\mathcal{D}_R}$,*

$$\prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{err on }(i,u,v)} Z(R,i,u,v,a)\right]\right) \geq p^{2\mathsf{Err}}. \tag{2.23}$$

Proof. First, we recall that edges in E' have the property that the parties share their random string correctly - hence, we have $R_{i,u,v} = R_{i,v,u}$ for all $(u,v) \in E'$ and iterations i.

We split Z into two subevents Z_1 and Z_2 in the natural way, such that $Z = Z_1 \wedge Z_2$. Formally:

$$Z_1(R,i,u,v,a) \overset{\text{def}}{=} \left\{1_{h_{R,i,u,v}(T^{(a)}_{u,v}(i))=\widetilde{h}_{R,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,u,v}\right\} \quad \text{and} \tag{2.24}$$
$$Z_2(R,i,u,v,a) \overset{\text{def}}{=} \left\{1_{\widetilde{h}_{R,i,u,v}(T^{(a)}_{u,v}(i))=h_{R,i,u,v}(T^{(a)}_{v,u}(i))} = a_{i,v,u}\right\} \tag{2.25}$$

where we also used the fact that $R_{i,u,v} = R_{i,v,u}$ in the above definitions.

Since a is consistent with respect to $\mathcal{D}_R$, i.e., $\mathbb{P}[A = a] > 0$, then by Proposition 2.3.19 there must exist R^* such that $\bigwedge_{i,u,v} Z(R^*,i,u,v,a)$ holds. In other words, for this specific R^*, we have that for all (i,u,v) such that $(u,v) \in E'$ and iteration i, we have that $Z_1(R^*,i,u,v,a) = Z_2(R^*,i,u,v,a) = 1$. If o_1 and o_2 are the hash outputs on the link (u,v) in iteration i with randomness R^* (i.e. $o_1 = h_{R^*,i,u,v}(T_{u,v}^{(a)})$ and $o_2 = h_{R^*,i,u,v}(T_{v,u}^{(a)})$), then if $a_{i,u,v} = 1$, the noise exactly makes $\widetilde{h}_{R,i,u,v}(T_{v,u}^{(a)}(i)) = o_1$, and otherwise it does not. Similarly, if $a_{i,v,u} = 1$ then the noise makes $\widetilde{h}_{R,i,u,v}(T_{u,v}^{(a)}(i)) = o_2$, and otherwise it does not. We will say that o_1 and o_2 make the event Z (resp. Z_1, Z_2) *occur*, when the iteration i, link (u,v), and agreement pattern a are understood from context.

Fix an iteration i, link (u,v), and agreement pattern a. Assume $T_{u,v}^{(a)} = T_{v,u}^{(a)}$. Then, if there is no noise in the message $h_{R,i,u,v}(T_{v,u}^{(a)}(i))$ it must be that $a_{i,u,v} = 1$, since we know that the hashes will match even if the random string used were $R_{i,u,v} = R^*_{i,u,v}$. In fact, *any* seed $R_{i,u,v}$ makes the hashes match, and so since $a_{i,u,v} = 1$, we conclude that $\mathbb{P}_R[Z_1(R,i,u,v,a)] = 1$.

If there is noise in that message, then it must be that $a_{i,u,v} = 0$ since for any seed $R_{i,u,v}$ (including $R^*_{i,u,v}$), the hashes will not match, since they matched before the noise and the noise makes one of the messages different. Once again, since this holds for all possible random strings $R_{i,u,v}$, we have that $\mathbb{P}_R[Z_1(R,i,u,v,a)] = 1$ holds for this case as well.

Now assume that $T_{u,v}^{(a)} \neq T_{v,u}^{(a)}$. Furthermore, assume there is no noise in the message $h_{R,i,u,v}(T_{v,u}^{(a)}(i))$. If $a_{i,u,v} = 1$, then, any seed R that causes a collision will make $Z_1(R,i,u,v,a) = 1$. I.e.,

$$\mathbb{P}_R[Z_1(R,i,u,v,a)] = \mathbb{P}_R[h_{R,i,u,v}(T_{u,v}^{(a)}(i)) = h_{R,i,v,u}(T_{v,u}^{(a)}(i))] = p,$$

where the last inequality uses the fact that p is *exactly* the collision probability of the inner product hash function we use (Lemma 2.1.3). If, on the other hand, $a_{i,u,v} = 0$, then by a similar argument,

$$\begin{aligned}\mathbb{P}_R[Z_1(R,i,u,v,a)] &= \mathbb{P}_R[h_{R,i,u,v}(T_{u,v}^{(a)}(i)) \neq h_{R,i,u,v}(T_{v,u}^{(a)}(i))] \\ &= 1 - \mathbb{P}_R[h_{R,i,u,v}(T_{u,v}^{(a)}(i)) = h_{R,i,u,v}(T_{v,u}^{(a)}(i))] \\ &\geq 1 - p.\end{aligned}$$

Next let us assume that there was noise in the message $h_{R,i,u,v}(T_{u,v}^{(a)}(i))$. To be concrete, we will denote the additive noise by x.

If $a_{i,u,v} = 1$, then, since $Z_1(R^*, i, u, v, a) = 1$, we know that $o_1 = \widetilde{o_2} = o_2 + x$, where we recall that o_1 and o_2 are defined to be $h_{R^*,i,u,v}(T^{(a)}_{u,v}(i))$ and $h_{R^*,i,u,v}(T^{(a)}_{v,u}(i))$ respectively. In fact, we can conclude that for any random string $R_{i,u,v}$ such that $h_{R,i,u,v}(T^{(a)}_{u,v}(i)) = h_{R,i,u,v}(T^{(a)}_{v,u}(i)) - x$, will make $Z_1(R, i, u, v, a) = 1$.

Since the hash-function h is based on inner-product (Definiton 2.1.2), its output is uniform when the seed is uniformly distributed and the input is nonzero (Lemma 2.1.3), and we get

$$\begin{aligned}\mathbb{P}_R[Z_1(R, i, u, v, a)] &\geq \mathbb{P}_R[(h_{R,i,u,v}(T^{(a)}_{u,v}(i)) = h_{R,i,u,v}(T^{(a)}_{v,u}(i)) - x] \\ &= p.\end{aligned}$$

If, on the other hand, $a_{i,u,v} = 0$, then

$$\begin{aligned}\mathbb{P}_R[Z_1(R, i, u, v, a)] &= \mathbb{P}_R[h_{R,i,u,v}(T^{(a)}_{u,v}(i)) \neq \widetilde{h}_{R,i,u,v}(T^{(a)}_{v,u}(i))] \\ &= 1 - \mathbb{P}_R[h_{R,i,u,v}(T^{(a)}_{u,v}(i)) = h_{R,i,u,v}(T^{(a)}_{v,u}(i)) - x] \\ &\geq 1 - p\end{aligned}$$

where again this last transition follows due to the uniform output of the inner product hash function (Lemma 2.1.3).

Hence, in all cases we have

$$\mathbb{P}_R[Z_1(R, i, u, v, a)] \geq p.$$

Similar arguments apply to $Z_2(R, i, u, v, a)$. Since both events are independent when the transcripts are different (they use different seeds for the hashes, and the transcript is fixed by a and the noise pattern), we get that for any appropriate (i, v, u),

$$\mathbb{P}_R[Z(R, i, u, v, a)] \geq p^2.$$

Recall that the seeds $R_{i,u,v}$ are independent and uniform when $(u,v) \in E'$. Since there are Err triples (i,u,v) such that an error occurs in (i,u,v), we conclude that

$$\prod_{(u,v)\in E'} \mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{err on }(i,u,v)} Z(R,i,u,v,a)\right] \geq p^{2\mathsf{Err}}.$$

□

Remark 2.3.23 (Remark on Fixing Adversaries). *In Section 2.2, we proved that our scheme is resilient to both additive adversaries and* fixing *adversaries. The latter type of adversary commits errors where it fixes the communication sent on a link in a given round. In the proof of Claim 2.3.22 we have assumed the adversary is additive; however, it can easily be verified that the claim also holds for fixing adversaries. The main idea is that, for any symbol $\widetilde{o_2}$ that the adversary puts on a link (which could also be silence), the probability that $h_{R,i,u,v}(T^{(a)}_{u,v}(i)) = \widetilde{o_2}$ is exactly equal to p due to the uniform output distribution of the hash. This is the only step of the proof which differs for fixing and additive adversaries.*

Claim 2.3.22 along with Eq. (2.17) lead to Eq. (2.15):

$$\begin{aligned}
&\sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{S\sim\mathcal{D}_S}[A=a]\\
&\leq e\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i:\text{no err on }(i,u,v)} Z(R,i,u,v,a)\right]\right)\cdot Q(a)\\
&\overset{(1)}{\leq} e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i} Z(R,i,u,v,a)\right]\right)\cdot Q(a)\\
&\overset{(2)}{\leq} e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i} Z(R,i,u,v,a)\right]\right)\cdot \mathbb{P}_{S\sim\mathcal{D}_S}\left[\bigwedge_{i,(u,v)\in E\setminus E'} Z(S,i,u,v,a)\right]\\
&\overset{(3)}{\leq} e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \prod_{(u,v)\in E'} \left(\mathbb{P}_{R_{u,v}\sim\mathcal{U}}\left[\bigwedge_{i} Z(R,i,u,v,a)\right]\right)\cdot \mathbb{P}_{S\sim\mathcal{D}_R}\left[\bigwedge_{i,(u,v)\in E\setminus E'} Z(S,i,u,v,a)\right]\\
&\overset{(4)}{=} e\cdot p^{-2\mathsf{Err}}\cdot \sum_{a\in\mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{R\sim\mathcal{D}_R}[A=a].
\end{aligned}$$

Transition (1) is via Claim 2.3.22. (2) is by the definition of Q (Eq. (2.16)). (3) uses the fact that $\mathcal{D}_R$ and $\mathcal{D}_S$ are identical for $(u,v) \in E \setminus E'$. (4) follows from Proposition 2.3.19.

Hence, we have proved the desired relation between $\sum_{a \in \mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{S\sim\mathcal{D}_S}[A = a]$ and between $\sum_{a \in \mathsf{Bad}_{\mathcal{D}_R}} \mathbb{P}_{R\sim\mathcal{D}_R}[A = a]$ as stated in Eq. (2.15). We recall that this implies that $\mathbb{P}_{S\sim\mathcal{D}_S}[D > k \cdot \mathsf{Err}]$ and $\mathbb{P}_{R\sim\mathcal{D}_R}[D > k \cdot \mathsf{Err}]$ have the same relation, which concludes the the proof of Lemma 2.3.13. □

We now show that even with arbitrary oblivious tampering of random strings on the edges $E \setminus E'$, the adversary cannot cause too many dangerous triples when the seeds $R_{u,v}$ are uniform for $(u,v) \in E'$. Let $X_{i,u,v}$ be an indicator random variable for the event that there is a hash collision between u and v in iteration i. Define $\psi_{u,v}$ as follows.

Algorithm 9 The process $\psi_{u,v}$

```
$i \leftarrow 1, \psi_{u,v}(1) \leftarrow 0$
**for all** iterations $i$ from 1 to $100|\Pi|$ **do**
    **if** error occurs between $u$ and $v$ during iteration $i$, during any phase **then**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + 6C_6$
    **else if** $\varphi_{u,v}(i) > 0$ **then**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + 5C_6 X_{i,u,v} - 5(1 - X_{i,u,v})$
    **else**
        $\psi_{u,v}(i+1) = \psi_{u,v}(i)$
```

Let D^* denote the set of triples (i,u,v) where $\psi_{u,v}(i) > 0$ and $(u,v) \in E'$. We proved in Section 2.2 that $\psi_{u,v} \geq \varphi_{u,v}$ for all iterations, which we recall now.

Lemma 2.3.24 (Lemma 2.2.15 in Section 2.2)**.** *For all iterations $i \in [100|\Pi|]$ and all edges $(u,v) \in E'$, we have that $\psi_{u,v}(i) \geq \varphi_{u,v}(i)$, where $\varphi_{u,v}(i)$ denotes the value of the potential $\varphi_{u,v}$ at the beginning of iteration i.*

It follows from Lemma 2.2.15 that $|\mathsf{D}^*| \geq D$.

We now proceed to upper bound $|\mathsf{D}^*|$ with high probability, just like in Section 2.2.

Lemma 2.3.25. *Suppose the adversary is oblivious, and let* Err *be the number of errors committed during this part. Let* $p = 2^{-\Theta(K/m)}$ *be the hash collision probability of* h*, and assume that* $p < \frac{1}{30C_6}$. *Let* k *be some real number such that* $k \geq 10C_6$. *Let* D^* *denote the set of triples* (i,u,v) *such that* $\psi_{u,v}(i) > 0$ *and* $(u,v) \in E'$. *Then*

$$\mathbb{P}_R[|\mathsf{D}^*| > k\mathsf{Err}] < p^{\Omega(k\cdot\mathsf{Err})},$$

where R *is defined as in Lemma 2.3.13, and the* Ω *hides constants on the order of* $1/C_6$.

Proof. Consider an instance of the protocol after fixing the adversary's errors, and assuming uniformly random seeds for the hashes. Assume $|\mathsf{D}^*| > k \cdot \mathsf{Err}$ and let $\widetilde{\mathsf{D}^*} \subseteq \mathsf{D}^*$ denote the set of triples (i,u,v) such that $\psi_{u,v}(i) > 0$ and $(u,v) \in E'$, and no error occurs between u and v in iteration i. Since the number of errors is Err, we get that $|\widetilde{\mathsf{D}^*}| \geq (k-1)\mathsf{Err}$. Define

$$\psi \overset{\text{def}}{=} \sum_{(u,v)\in E'} \psi_{u,v}.$$

We know that ψ is always nonnegative by design, since each of the $\psi_{u,v}$'s are nonnegative. Consider $\psi(100|\Pi|+1)$, that is, the value of ψ immediately after the final $100|\Pi|$-th iteration of Algorithm A. Recall that $\psi_{u,v}$ goes up by $6C_6$ whenever there is an error between u and v, goes up by $5C_6$ whenever there is a hash collision between u and v, and goes down by 5 whenever there is no hash collision between u and v and $\psi_{u,v}$ was positive. This yields

$$0 \leq \psi(100|\Pi|+1) \leq 6C_6 \cdot \mathsf{Err} + \sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} (5C_6 X_{i,u,v} - 5(1 - X_{i,u,v})), \tag{2.26}$$

Rearranging, we note that

$$\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq \frac{5|\widetilde{\mathsf{D}^*}| - 6C_6 \cdot \mathsf{Err}}{5C_6+5}$$
$$\geq \frac{4}{5C_6+5}|\widetilde{\mathsf{D}^*}|$$

where the last inequality follows from $|\widetilde{\mathsf{D}^*}| \geq (k-1)\cdot\mathsf{Err} \geq 6C_6\cdot\mathsf{Err}$. So we can upper bound $\mathbb{P}_R[|\mathsf{D}^*| > k\mathsf{Err}]$ by upper bounding the probability that $\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq \frac{4}{5C_6+5}|\widetilde{\mathsf{D}^*}|$. Note that the events of hash collisions on the links E' are independent, due to the fact that the adversary fixes their errors before the seeds are sampled. Furthermore, they all happen with at most probability p. So, by a Chernoff bound, we get that

$$\mathbb{P}\left[\frac{1}{|\widetilde{\mathsf{D}^*}|}\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \geq \frac{4}{5C_6+5} \geq p + \frac{3}{5C_6+5}\right] \leq p^{\Omega(|\widetilde{\mathsf{D}^*}|)} \leq p^{\Omega(k\cdot\mathsf{Err})} \tag{2.27}$$

□

Corollary 2.3.26 (Corollary of Lemmas 2.3.25 and 2.3.13). *Suppose $|\Pi|K \geq m\log(m)$, and the adversary is oblivious and makes* Err *errors. Let k be a constant that is sufficiently larger than C_6 and the constant term in the exponent of p, i.e. $\frac{\log(1/p)}{K/m}$. Suppose that we run an instance of Algorithm A and denote the random string sampled by the protocol as S. Let $E' \subseteq E$ and D be as in Lemma 2.3.13. Then*

$$\mathbb{P}_S[D \geq k\cdot\mathsf{Err}] \leq p^{\Omega(k\mathsf{Err})} = \exp(-\Omega(k\mathsf{Err}\cdot(K/m))$$

Proof. We apply Lemma 2.3.13 and use the fact that $|\Pi|K \geq m\log(m)$, we have that

$$\mathbb{P}_S[D \geq k\cdot\mathsf{Err}] \leq e\cdot p^{-2\mathsf{Err}}\cdot\mathbb{P}_R[D \geq k\cdot\mathsf{Err}]$$

where R is uniform for seeds on the edges in E'. Recall that $|\widetilde{\mathsf{D}^*}| \geq |\mathsf{D}^*|$, and so applying

Lemma 2.3.25 we get that

$$\mathbb{P}_S[D \geq k \cdot \mathsf{Err}] \leq e \cdot p^{-2\mathsf{Err}} \cdot p^{\Omega(k\mathsf{Err})} = p^{\Omega(k\mathsf{Err})}.$$

□

Bounding the communication We continue to bound the communication in Algorithm A. We start with an easy claim bounding the communication that occurs during the randomness exchange.

Claim 2.3.27. *Assume that* $|\Pi|K \geq m\log(m)$*. Then the communication in the randomness exchange (Algorithm 7) is* $\Theta(K|\Pi|)$*.*

Proof. Since $\delta = 2^{-\Theta(|\Pi|K/m)}$ (Algorithm 6), every link needs to exchange during this part a seed $L \in \{0,1\}^r$ of length

$$\begin{aligned} r &= \Theta\left(\log(1/\delta) + \log(\ell)\right) \\ &= \Theta(|\Pi|K/m)) + \Theta(\log(|\Pi|^2 K)) \\ &= \Theta\left(|\Pi|\frac{K}{m}\right). \end{aligned}$$

where in the last line we use the fact that $\log(K) = \Theta(\log(m))$ for all settings of K that we use (either m or $m\log(m)$), and that $|\Pi|K/m \geq \log(m)$.

Each such seed is encoded via an error correcting code with a constant rate, hence, each such codeword takes $\Theta(|\Pi|K/m))$ bits. Since there are m links in the network, the claim holds. □

Next we prove that the communication cannot get too high without a large number of errors. Note that some links have arbitrary, potentially disagreeing, random strings, and hence the corresponding parties might have hash-collisions in all iterations. While this can make the communication very large, our goal is to show that the communication cannot become *too* large without having many dangerous triples (where again, dangerous triples only count on edges $(u,v) \in E'$). Once we establish this fact, we will bound the probability of having a large amount of dangerous triples.

Towards this end, we first recall that Lemma 2.3.10 connects the number of errors and hash collisions in a specific iteration to the potential increase and the communication that these errors and collisions might result in the same iteration.

We now prove the connection between the large communication and large amount of dangerous rounds (assuming the noise level is within the allowed threshold).

Lemma 2.3.28. *Fix $\gamma \geq 2$. Denote the number of errors in the "main" part of the protocol as* Err, *and the communication in this part as* CC. *Let D and E' be the same as in Lemma 2.3.13. Then* $\mathsf{CC} \geq 100\gamma\alpha|\Pi|K(1+|E \setminus E'|)$ *and* $\mathsf{Err} \leq \frac{\varepsilon \mathsf{CC}}{K}$ *only if* $\mathsf{Err} > 0$ *and* $D > \beta \cdot \mathsf{Err}$, *where* $\beta \geq \max\left(\frac{1}{3\alpha\varepsilon}, \frac{33\gamma|\Pi|(1+|E\setminus E'|)}{\mathsf{Err}}\right)$*, and α is the constant multiplying the communication in Lemma 2.3.10.*

Proof. By Lemma 2.3.10, we can bound

$$\mathsf{CC} \leq \alpha K(100|\Pi| + \mathsf{EHC}) \tag{2.28}$$

A "hash collision" (the HC in EHC) occurs when two hashes that are passed on an edge match, but the transcripts do not match. Note that on any edge $(u,v) \in E \setminus E'$, the seeds may not have even been shared correctly! We bound the number of "hash collisions" on edges like this trivially, by $100|\Pi|$. Note that if u and v have different seeds, their hash outputs may disagree even when their transcripts are the same. While this is problematic for simulating the protocol correctly, these sorts of "fake" hash misses do not increase the communication, since this stops parties from talking rather than making them talk too much (unlike hash collisions, which can make parties talk excessively, as seen in Lemma 2.2.9.

By noting that $\mathsf{Err} \leq \frac{\varepsilon}{K} \cdot \mathsf{CC}$ and that the number of hash collisions on links in E' is upper bounded by the number of dangerous triples D, we can rewrite Eq. (2.28) as

$$\begin{aligned}\mathsf{CC} &\leq \alpha K\left(100|\Pi| + \frac{\mathsf{CC}\cdot\varepsilon}{K} + D + HC_{E\setminus E'}\right)\\ &\leq \alpha K\left(100|\Pi| + \frac{\mathsf{CC}\cdot\varepsilon}{K} + D + 100|\Pi|\cdot|E\setminus E'|\right)\end{aligned}$$

Rearranging and using the assumption that $\mathsf{CC} \geq 100\gamma\alpha|\Pi|K(1+|E\setminus E'|)$ and $\gamma \geq 2$, we get that

$$\begin{aligned}D &\geq \frac{(1-\alpha\varepsilon)\mathsf{CC} - 100|\Pi|\alpha K(1+|E\setminus E'|)}{\alpha K}\\ &\geq \frac{\mathsf{CC}}{3\alpha K}\\ &\geq \beta\mathsf{Err}\end{aligned}$$

Note that if $\mathsf{Err} = 0$, then specifically there are no errors on the link in E' in the main part of the protocol. Since the parties agree on seeds in E', this means there will be no dangerous triples, and hence D cannot be larger than $\frac{\mathsf{CC}}{3\alpha K}$. The bounds on β follow since $\mathsf{CC} \geq \frac{K}{\varepsilon}\mathsf{Err}$ and $\mathsf{CC} \geq 100\gamma\alpha|\Pi|K(1+|E\setminus E'|)$ respectively. □

Now we can prove our main communication bound of the section.

Lemma 2.3.29. *Assume that $|\Pi|K \geq m\log(m)$. Denote the number of errors in the "main" part of the protocol as* Err*, and denote the communication complexity in the main part as* CC*. Fix $\varepsilon > 0$. Let D and E' be the same as in Lemma 2.3.13. Suppose that the tuple of random strings S used are as in the randomness exchange (Algorithm 7), and the adversary tampered with strings on links $E\setminus E'$. Fix $\gamma \geq 2$. The probability that the communication complexity of Algorithm A is* $\mathsf{CC} \geq 100\gamma\alpha|\Pi|K(1+|E\setminus E'|)$ *and* $\mathsf{Err} \leq \frac{\varepsilon}{K}\cdot\mathsf{CC}$ *is at most*

$$p^{\Omega(\gamma|\Pi|\cdot(1+|E\setminus E'|))}$$

where α is the constant multiplying the communication in Lemma 2.3.10.

Proof. Suppose that $\mathsf{CC} \geq 100\gamma\alpha|\Pi|K(1+|E\setminus E'|)$ and $\mathsf{Err} \leq \frac{\varepsilon}{K}\cdot\mathsf{CC}$. Then by applying

Lemma 2.3.28, we have that $D \geq \beta\mathsf{Err}$, where β is as defined in Lemma 2.3.28. Since $\beta \geq \frac{1}{3\alpha\varepsilon} > 20C_6$ by taking ε sufficiently small, we apply Corollary 2.3.26 to get that the probability of this is at most

$$p^{\Omega(\beta\mathsf{Err})}$$

Since $\beta \geq \frac{33\gamma|\Pi|(1+|E\setminus E'|)}{\mathsf{Err}}$, we conclude that the probability of this event is at most

$$p^{\Omega(\gamma|\Pi|\cdot(1+|E\setminus E'|))}$$

□

Proof of Theorem 2.3.9 Recall that we have $K = m$ in Algorithm A, and so we will set $K = m$ throughout this proof. We split the proof into three steps. First, we show that the probability that $|E \setminus E'| \geq 1$ and $\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}$ is at most $\exp(-\Omega(|\Pi|))$. Then, we set $E' = E$ and show that the probability that $\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}$ and $\mathsf{CC} > \Theta(|\Pi|m)$ is also exponentially small. Finally, we show that the probability that $\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}$, $\mathsf{CC} \leq \Theta(|\Pi|m)$, and $D \geq \omega(\varepsilon|\Pi|)$ is also exponentially small. This covers the space of bad events for our protocol; outside of these events, it can be shown that the protocol is simulated correctly, by following the proof of Theorem 2.2.2.

Denote the number of errors in the main part of the protocol as Err', and the communication in the main part as CC'.

Claim 2.3.30.

$$\mathbb{P}\left[\left(\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}\right) \wedge (|E \setminus E'| \geq 1)\right] < \exp(-\Omega(|\Pi|)).$$

Proof. Note that if $|E \setminus E'| \geq 1$, then the adversary must have committed $\Omega(|E \setminus E'| \cdot |\Pi|)$ errors in the randomness exchange phase. This holds since we encode each random strings with an error-correcting code with length $r = \Theta(|\Pi|)$ (Claim 2.3.27) and constant distance. The code having constant distance means that in order to corrupt even a single codeword, the adversary must make at least $\Theta(|\Pi|)$ errors. But since $\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}$, such an attack is within the adversary's budget only if $\mathsf{CC} \geq \frac{m}{\varepsilon}|\Pi| \cdot |E \setminus E'|$.

Note that the communication in the randomness exchange phase is at most $\Theta(|\Pi|m)$, since there are m links and strings of length $\Theta(|\Pi|)$ are sent on each. Hence, the amount of communication in the main part of the protocol must be at least

$$\mathsf{CC}' \geq \Theta\left(\frac{m}{\varepsilon}|\Pi| \cdot |E \setminus E'|\right) \geq 200\alpha|\Pi|m(1 + |E \setminus E'|),$$

where we use the assumption that $|E \setminus E'| \geq 1$.

By Claim 2.3.14, the error in the main part satisfies $\mathsf{Err}' \leq \frac{\varepsilon'}{m}\mathsf{CC}'$ for $\varepsilon' = \Theta(\varepsilon)$. So we can apply Lemma 2.3.29 with $\gamma = 2$ and allowed noise rate (for the main part) of ε'/m, and get that the probability that $\mathsf{CC}' \geq 200\alpha|\Pi|m(1 + |E \setminus E'|)$ is at most $\exp(-\Omega(|\Pi| \cdot (1 + |E \setminus E'|))) \leq \exp(-\Omega(|\Pi|))$, as stated. □

Given the above claim, we know that the adversary cannot corrupt even a single seed and keep its attack within the allowed budget. Hence, we can assume from this point on that $E' = E$, and that all the seeds are δ-biased strings chosen by the parties. Under these conditions, Lemma 2.3.29 gives us that

$$\mathbb{P}\left[(\mathsf{CC}' \geq 200\alpha|\Pi|m) \wedge \left(\mathsf{Err}' \leq \frac{\varepsilon'}{m}\mathsf{CC}'\right)\right] < \exp(-\Omega(|\Pi|)).$$

Now further suppose that the communication in the main part satisfies $\mathsf{CC}' \leq 200\alpha|\Pi|m$ and that $\mathsf{Err}' \leq \frac{\varepsilon'}{m}\mathsf{CC}'$. This implies that the number of errors the adversary commits is at most $200\alpha\varepsilon'|\Pi|$. Combining this with Corollary 2.3.26, we see that the probability that the adversary can cause the number of dangerous triples exceed $k\varepsilon'|\Pi|$ in this case is at most $\exp(-\Omega(k\varepsilon'|\Pi|))$, even when the parties share δ-biased randomness, where the $k \geq 10C_6$. Hence,

Corollary 2.3.31. *For a real number k such that $k \geq 10C_6$, we have that*

$$\mathbb{P}[\mathsf{EHC} > (k+1)(200\alpha\varepsilon')|\Pi|] < \exp(-\Omega(k\varepsilon'|\Pi|)).$$

To conclude, if we limit the adversary to corrupting at most $\mathsf{Err} \leq \frac{\varepsilon}{m}\mathsf{CC}$ transmissions, then the probability that the adversary is able to tamper with any of the of the randomness exchanges, cause more than $\Theta(|\Pi|m)$ communication throughout the protocol, or cause the number of errors and hash collisions to exceed $\Theta(k\varepsilon'|\Pi|)$, is at most $\exp(-\Omega(k\varepsilon'|\Pi|))$, where we have freedom to set k to be as large as we would like.

Regardless of the seed being used, Lemma 2.3.10 shows that Algorithm A concludes with $\phi \geq 100|\Pi|m$. Then, as long as none of the above bad events happen, we can take ε (and therefore also ε') to be sufficiently small, which (by applying Corollary 2.3.31 with appropriately selected k) makes EHC at most $|\Pi|/C_7$ with probability at least $1 - \exp(|\Pi|)$. Finally, we appeal to Claim 2.3.32, a reformulation of Claim 2.2.19 from Section 2.2, to conclude that Algorithm A correctly simulates Π.

Claim 2.3.32 (Reformulation of Claim 2.2.19)**.** *Suppose that ϕ is defined as in Equation 2.6, that $\phi \geq 100|\Pi|K$, and that $\mathsf{EHC} \leq \varepsilon^*|\Pi|$ for a sufficiently small constant $\varepsilon^* > 0$ that does* not *go to 0 as ε goes to 0. Then the underlying protocol Π has been simulated correctly.*

To conclude the proof, we note that EHC is bounded by $\varepsilon^*|\Pi|$ with probability $1 - \exp(-\Omega(|\Pi|))$ for Algorithm A follows from Corollary 2.3.31 by taking ε' to be sufficiently small with respect to ε^*, where the Ω hides a factor of the constant ε^* (but not ε or ε'). For more details, see the proof of Theorem 2.2.2; the proof follows identically in this case.

2.3.4 Coding Scheme for Non-Oblivious Noise

Overview

In this section, we generalize the result of the previous section to hold for nonoblivious adversaries. Nonoblivious adversaries have a lot of power: specifically, they can look at the random seeds that will be used in the rest of the protocol and choose their errors as a function of these seeds. The upshot is that we can no longer guarantee that the event of a hash collision is independent of previous hash collisions, or even close to independent. We briefly recall (from the Introduction to Section 2.3.1) the issues that can occur with an adversary that knows the random seeds used ahead of time, even when these seeds are initially sampled from a uniform distribution.

Suppose that the seeds to the hash functions are fixed, and that a randomly selected error creates a hash collision in the next round with probability p. Then roughly a p fraction of the possible errors would lead to a hash collision on this link in the next Simulation phase, and hence be undiscovered. Among the errors that caused a collision, roughly a p fraction of them would lead to a hash collision in the proceeding link, and so on. Hence, if the number of error patterns is sufficiently large, the adversary can select a specific error pattern out of them that will not be detected within a large sequence of iterations. In particular, since there are $K = \text{poly}(m)$ bits communicated in a single chunk, the adversary has a choice of $\text{poly}(m)$ errors in any simulation phase, so that there exists an error that is undetectable for $\Theta(\log(m))$ iterations with high probability.

To address this issue, we modify our hash function to have a collision probability of $p = \frac{1}{m^{\Theta(1)}}$, by increasing the output size of our hash to $\Theta(\log(m))$ bits. This, however, affects the rate of the coding scheme. In order to keep the rate constant, we set K to be $m\log(m)$, i.e., the size of each chunk increases and we effectively send hashes (and other meta-data) less frequently. We then appeal to the strategy of [70], and show that there is no *oblivious, additive* adversary that sabotages the simulation with high probability. In particular, for any given oblivious, additive adversary, there are exponentially few "bad" randomness strings that do not guarantee a correct and constant rate simulation. Taking a union bound over all possible oblivious, additive adversaries, we show that the union of all the "bad" randomness strings is still exponentially small. This means that unless a bad randomness is picked by the parties, *no* oblivious, additive adversary can invalidate the simulation. But the actions of any non-oblivious adversary that makes Err errors can always be modeled by *some* oblivious, additive adversary that makes Err errors - simply take the oblivious adversary that miraculously manages to make the same corruptions. Hence, no non-oblivious adversary can invalidate the simulation for the same randomness.

Noise-resilient simulation for non-oblivious adversarial noise

Define Algorithm B to be Algorithm 1 combined with the InitializeState() procedure described in Algorithm 10 below.

Algorithm 10 InitializeState() for non-oblivious noise without a CRS

```
 1: $K \leftarrow m\log(m)$
 2: **for all** neighbors $v \in \mathcal{N}(u)$ in parallel **do**
 3:     Initialize $T_{u,v} = \emptyset$
 4:     Initialize $k_{u,v}, E_{u,v}, mpc1_{u,v}, mpc2_{u,v} \leftarrow 0$
 5:     $status_{u,v} \leftarrow$ "simulate"
 6:     $alreadyRewound_{u,v} \leftarrow 0$
 7:     $\delta := 2^{-\Theta(|\Pi|K/m)}$
 8:     $S := (S_{i,u,v})_{i\in[100|\Pi|]} \leftarrow$ RandomnessExchange($u$, $v$, $\delta$, $K$)
 9: $status_u \leftarrow 1$.
10: $netCorrect_u \leftarrow 1$
```

The only difference in InitializeState() from Section 2.3.3 is that K is set to $m\log(m)$.

Algorithm B: Analysis

The main result of this section is the following theorem, arguing that Algorithm B is resilient to non-oblivious adversary that corrupts up to a fraction $\varepsilon/m\log m$ of the transmissions, with high probability.

Theorem 2.3.33. *Assume a network $G = (V, E)$ with $n = |V|$ parties and $m = |E|$ links. Suppose Π is a multiparty protocol over the network G with communication complexity $\mathsf{CC}(\Pi)$, binary alphabet and fixed order of speaking. Let $|\Pi| = \frac{\mathsf{CC}(\Pi)}{5m\log(m)}$ and let $\varepsilon > 0$ be a sufficiently small constant. Algorithm B correctly simulates Π with communication complexity $O(\mathsf{CC}(\Pi))$ with probability at least $1 - \exp(-\Omega(|\Pi|\log(m)))$ over the randomness of the parties, in the presence of a non-oblivious adversary limited to a noise fraction at most $\varepsilon/m\log(m)$.*

Recall that in Algorithm B we set $K = m\log(m)$. We will use K and $m\log(m)$ interchangably throughout this section, often leaving expressions in terms of K and plugging in $m\log(m)$ at the end.

As mentioned in the overview in Section 2.3.4, our general strategy is to union bound over all oblivious additive adversaries to show that the fraction of random strings that lead to bad outcomes is small. We start by showing a simple but useful claim, which provides a bound on the number of different oblivious, additive adversaries that can commit a fixed number of errors.

Claim 2.3.34. *Fix a nonnegative integer* Err. *Assuming a large enough* m*, the number of different oblivious additive adversaries that commit exactly* Err *errors is at most*

$$\binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}} \cdot 2^{\mathsf{Err}}$$

for some constant $C > 0$*. Furthermore, if* $\mathsf{Err} > C\varepsilon m|\Pi|$*, then* $\mathsf{Err} > \frac{\varepsilon}{m\log(m)}\mathsf{CC}$*, where* C *is the same constant and* CC *denotes the communication of the robust protocol.*

Proof. In the randomness exchange part of the protocol, there are $\Theta(\frac{K}{m}|\Pi|) = \Theta(\log(m)|\Pi|)$ rounds, and in each of them there are at most m links which the adversary can corrupt. Therefore, there are $\Theta(m\log(m)|\Pi|) = O(m^2\log(m)|\Pi|)$ places for the adversary to put errors in the randomness exchange phase.

In the main part of the protocol, each iteration has at most $O(m\log(m))$ rounds, there are $100|\Pi|$ iterations, and in each round, the adversary can choose which of $2m$ possible (directed) links to put an error. Putting this together, we get that there are at most $O(m^2\log(m)|\Pi|)$ places for the adversary to put an error in the main part of the protocol. The first part of the claim follows by just letting C be the constant hidden in the big O.

We now justify why there are at most $O(m\log(m))$ rounds per iteration. Each meeting points phase takes $\Theta(K/m)$ rounds in which 5 hash values, each of length $\Theta(K/m)$, are being exchanged between each pair of parties (in parallel). Note that $\Theta(K/m) = \Theta(\log m) = O(m\log m)$. The flag passing phase has $O(n) = O(m\log(m))$ rounds. The simulation phase takes $5m\log(m) + 1$ rounds (where one round is for the $\perp$ symbols), and the rewind phase takes $n \ll m\log m$ rounds. Putting them all together the round number is at most $O(m\log m)$.

In each place where an oblivious adversary commits an error, it can choose one of two possibilities (either bit flip or deletion when there is communication; or inserting a 0 or 1 when there is no honest transmission), which we formalized with an additive noise model. This yields the 2^{Err} term.

The final claim follows since there are only $\leq Cm^2 \log(m)|\Pi|$ pairs of (round, directed link) where there can possibly be communication during the protocol, and so we get a trivial upper bound on the communication complexity as $Cm^2 \log(m)|\Pi|$. □

Next, we consider the communication complexity of Algorithm B. Recall that while the round complexity is fixed, the communication complexity depends on the inputs, noise, and randomness. We now show that the probability that the adversary can make the communication of the robust protocol larger than $O(CC(\Pi))$, if bounded to its allowed budget, is negligible.

Recall that Algorithm B has two parts: the randomness exchange part, which happens in InitializeState(), and the "main" part, where the actual simulation occurs. Claim 2.3.27 bounds the communication in the randomness exchange by $\Theta(|\Pi|K) = \Theta(|\Pi|m\log(m))$. Therefore, we are only left to show that communication in the main part of Algorithm B is bounded by $\Theta(|\Pi|K)$.

Lemma 2.3.35. *Let* $|\Pi| = \frac{\mathsf{CC}(\Pi)}{5m\log(m)}$ *and let* $\varepsilon > 0$ *be a sufficiently small constant. The "main" part of Algorithm B has at most* $400\alpha|\Pi|m\log(m) = O(\mathsf{CC}(\Pi))$ *communication with probability at least* $1 - \exp(-\Omega(|\Pi|\log(m)))$ *over the randomness of the parties, in the presence of a non-oblivious adversary with noise fraction at most* $\varepsilon/m\log(m)$*, and where* α *is the constant in Lemma 2.3.10.*

This lemma immediately leads to a bound on the communication of the entire protocol as stated by the following corollary.

Corollary 2.3.36. *Assume the setting of Theorem 2.3.33. Algorithm B has at most* $O(|\Pi|m\log(m)) = O(\mathsf{CC}(\Pi))$ *communication with probability at least* $1 - \exp(-\Omega(|\Pi|\log(m)))$ *over the randomness of the parties, in the presence of a non-oblivious adversary with noise fraction at most* $\varepsilon/m\log(m)$.

Proof. By Claim 2.3.27, the communication in the randomness exchange phase is $O(|\Pi|K) = O(|\Pi|m\log(m))$, regardless of the randomness of the parties. Lemma 2.3.35 additionally bounds the communication in the main part of the protocol by $O(|\Pi|m\log(m))$ with probability $1-\exp(-\Omega(|\Pi|\log(m)))$ over the randomness of the parties. □

Finally, we upper bound the probability that the communication is too large for any oblivious adversary, with a bound that will be sufficient strong to union bound over all oblivious adversaries and prove Lemma 2.3.35.

Proposition 2.3.37. *Fix an oblivious adversary, and denote the number of errors it commits as* Err. *Let* CC *denote the communication complexity of the entire protocol, and* CC' *denote the communication complexity in the main part. Then*

$$\mathbb{P}\left[(\mathsf{CC}' > 400\alpha|\Pi|K) \wedge (\mathsf{Err} \le \tfrac{\varepsilon}{K}\mathsf{CC})\right] \le \begin{cases} \exp(-\Omega(|\Pi|\log(m))) & \text{if } \mathsf{Err} \le 400\alpha\varepsilon'|\Pi| \\ \exp(-\Omega(\log(m)\mathsf{Err}/\varepsilon)) & \text{otherwise} \end{cases}, \tag{2.29}$$

Proof. Recall from Section 2.3.3 that the adversary can tamper with the randomness exchanges on some links. In line with the notation in Section 2.3.3, denote the set of edges that share randomness successfully as $E' \subseteq E$, and additionally define $J \overset{\text{def}}{=} |E \setminus E'|$ to be the number of tampered edges, for notational convenience. Note that, since the randomness in the randomness exchange phase is encoded with a code with constant distance, and since $\log(1/\delta) = \Theta(K/m|\Pi|)$, we have that

$$J \le \Theta\left(\frac{\mathsf{Err}\cdot m}{K|\Pi|}\right) \tag{2.30}$$

This is because it takes $\Theta(\frac{K}{m}|\Pi|)$ errors to corrupt the randomness exchange on any single edge.

Furthermore, recall from Claim 2.3.14 that $\mathsf{Err} \le \frac{\varepsilon'}{K}\mathsf{CC}'$ for $\varepsilon' = \Theta(\varepsilon)$, and this implies that the *effective* noise rate on the main part of the protocol is at most ε'/K. Hence, we write

$$\mathbb{P}\left[(\mathsf{CC}' > 400\alpha|\Pi|K) \wedge \left(\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}\right)\right] \le \mathbb{P}\left[(\mathsf{CC}' > 400\alpha|\Pi|K) \wedge (\mathsf{Err} \le \frac{\varepsilon'}{K}\mathsf{CC}')\right]$$
$$\le \mathbb{P}\left[(\mathsf{CC}' > 400\alpha|\Pi|K) \wedge (\mathsf{CC}' \ge \frac{K}{\varepsilon'}\mathsf{Err})\right] \qquad (2.31)$$

where in the second line we just rearranged the condition that $\mathsf{Err} \le \frac{\varepsilon'}{K}\mathsf{CC}'$.

The fact that the effective noise rate is $\varepsilon' = \Theta(\varepsilon)$ on the main part also lets us invoke Lemma 2.3.29, which we now recall. This will be our main tool here.

$$\mathbb{P}\left[(\mathsf{CC}' > 100\gamma\alpha|\Pi|K(1+J)) \wedge (\mathsf{CC}' \ge \tfrac{K}{\varepsilon'}\mathsf{Err})\right] \le \exp(-\Omega(\gamma|\Pi| \cdot (1+J)\tfrac{K}{m})) \qquad (2.32)$$

where $\gamma \ge 2$ is a parameter we can set at our convenience.

We recall upper bound for J in terms of the number of errors, namely that $J \le \Theta(\frac{\mathsf{Err}\cdot m}{K|\Pi|})$. We can "plug in" this upper bound into Eq. (2.32) by absorbing a factor of $\frac{1+J}{1+\Theta(\mathsf{Err}\cdot m/(K|\Pi|))}$ into γ, and get that

$$\begin{aligned}
&\mathbb{P}\left[\left(\mathsf{CC}' > 100\gamma\alpha|\Pi|K\left(1+\Theta\left(\frac{\mathsf{Err}\cdot m}{K|\Pi|}\right)\right)\right) \bigwedge \left(\mathsf{CC}' \ge \frac{K}{\varepsilon'}\mathsf{Err}\right)\right] \\
&= \mathbb{P}\left[(\mathsf{CC}' > \gamma(100\alpha|\Pi|K + \Theta(\mathsf{Err}\cdot m))) \bigwedge \left(\mathsf{CC}' \ge \frac{K}{\varepsilon'}\mathsf{Err}\right)\right] \\
&\le \exp\left(-\Omega\left(\frac{\gamma|\Pi|K + \mathsf{Err}\cdot\gamma\cdot m}{m}\right)\right) \qquad (2.33)
\end{aligned}$$

where the above formula still holds for $\gamma \ge 2$, since we just made γ smaller by absorbing a factor less than 1.

We will apply Eq. (2.33) to bound the probability on the left-hand side of Eq. (2.29) for the two cases of the statement.

Case 1: $\mathsf{Err} \le 400\alpha\varepsilon'|\Pi|$.

In this case, we note that $\Theta(\mathsf{Err}\cdot m) \le 100\alpha|\Pi|K$ due to the fact that ε' can be taken to be much smaller than the constant hidden by the Θ. Hence, we can select $\gamma \ge 2$ such that

$400\alpha|\Pi|K \geq \gamma(100\alpha|\Pi|K + \Theta(\mathsf{Err} \cdot m))$. Therefore, Eq. (2.29) becomes as follows:

$$\begin{aligned}
&\mathbb{P}\Big[(\mathsf{CC}' > 400\alpha|\Pi|K) \wedge (\mathsf{CC}' \geq \tfrac{K}{\varepsilon'}\mathsf{Err})\Big] \\
&\leq \mathbb{P}\Big[(\mathsf{CC}' > \gamma(100\alpha|\Pi|K + \Theta(\mathsf{Err} \cdot m))) \wedge (\mathsf{CC}' \geq \tfrac{K}{\varepsilon'}\mathsf{Err})\Big] \\
&\leq \exp(-\Omega(|\Pi|\tfrac{K}{m})) \\
&\leq \exp(-\Omega(|\Pi|\log(m))) \qquad (2.34)
\end{aligned}$$

where $\gamma \geq 2$ is selected appropriately. The second line follows from Eq. (2.33) with the appropriately selected γ.

Case 2: $\mathsf{Err} > 400\alpha\varepsilon'|\Pi|$.

In this case, we claim that $\mathsf{CC}' \geq \frac{K}{\varepsilon'}\mathsf{Err}$ already implies that CC' is large enough to apply Eq. (2.33) to bound $\mathbb{P}[\mathsf{CC}' \geq \frac{K}{\varepsilon'}\mathsf{Err}]$. To see this, note that $\frac{\mathsf{CC}'}{2} \geq \frac{K}{2\varepsilon'}\mathsf{Err} \gg \Theta(\mathsf{Err} \cdot m)$ and that $\frac{\mathsf{CC}'}{2} \geq \frac{K}{2\varepsilon'}\mathsf{Err} \geq 200\alpha|\Pi|K$, which implies that $\mathsf{CC}' = \gamma(100\alpha|\Pi|K + \Theta(\mathsf{Err} \cdot m))$ for some $\gamma \geq 2$. Hence, we can apply Eq. (2.33) with appropriately selected $\gamma \geq 2$ to get that

$$\begin{aligned}
\mathbb{P}[\mathsf{CC}' \geq \tfrac{K}{\varepsilon'}\mathsf{Err}] &\leq \exp(-\Omega(\tfrac{K}{m\varepsilon'}\mathsf{Err})) \\
&\leq \exp(-\Omega(\log(m)\mathsf{Err}/\varepsilon)) \qquad (2.35)
\end{aligned}$$

□

Equipped with Proposition 2.3.37, we proceed to prove Lemma 2.3.35.

Proof of Lemma 2.3.35. For this proof, let CC' denote the communication complexity in the main part of Algorithm B.

We will union bound over all possible oblivious additive adversaries to show that none of them can make much communication. Recall from Claim 2.3.34 that the number of adversaries that commit exactly Err errors is at most

$$\binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}} \cdot 2^{\mathsf{Err}} \qquad (2.36)$$

and that we can take $\mathsf{Err} \le O(\varepsilon m|\Pi|)$ without loss of generality when considering adversaries that do not exceed their error budget.

Now suppose that $\mathsf{Err} \le 400\alpha\varepsilon'|\Pi|$. Since ε' is sufficiently small, Err is much smaller than $Cm^2\log(m)|\Pi|$, so we can apply the monotonicity of small binomial coefficients in their bottom argument to upper bound Eq. (2.36) by

$$\begin{aligned}
&\le \binom{Cm^2\log(m)|\Pi|}{400\alpha\varepsilon'|\Pi|}\cdot 2^{400\alpha\varepsilon'|\Pi|}. \\
&\le \left(\frac{Cem^2\log(m)|\Pi|}{400\alpha\varepsilon'|\Pi|}\right)^{400\alpha\varepsilon'|\Pi|}\cdot 2^{400\alpha\varepsilon'|\Pi|} \\
&\le \exp(O(\varepsilon|\Pi|\log(m))) && (2.37)
\end{aligned}$$

where the second transition is applying Eq. (1.1), which we recall says that $\binom{n}{k} \le (ne/k)^k$. Now we can use Proposition 2.3.37 to show that the total fraction of random strings that can allow *any* oblivious adversary that makes at most $400\alpha\varepsilon'|\Pi|$ errors to guarantee that $\mathsf{Err} \le \frac{\varepsilon}{K}\mathsf{CC}$ and that $\mathsf{CC} > 400\alpha|\Pi|m\log(m)$ is at most:

$$\begin{aligned}
&\sum_{\mathsf{Err}=0}^{400\alpha\varepsilon'|\Pi|}\binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}}\cdot 2^{\mathsf{Err}}\cdot\exp(-\Omega(|\Pi|\log(m))) \\
\le &\sum_{\mathsf{Err}=0}^{400\alpha\varepsilon'|\Pi|}\exp(O(\varepsilon|\Pi|\log(m)))\cdot\exp(-\Omega(|\Pi|\log(m))) \\
\le &\exp(-\Omega(|\Pi|\log(m))), && (2.38)
\end{aligned}$$

where we substitute Eq. (2.37) for the first transition.

Finally, we suppose that $\mathsf{Err} > 400\alpha\varepsilon'|\Pi|$. Then we again use Eq. (1.1) and Claim 2.3.34

to upper bound the number of oblivious adversaries that make exactly Err errors by

$$\begin{aligned}\binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}}\cdot 2^{\mathsf{Err}} &\leq \left(\frac{Cm^2\log(m)|\Pi|}{\mathsf{Err}}\right)^{\mathsf{Err}}\cdot 2^{\mathsf{Err}}\\ &\leq \left(\frac{Cm^2\log(m)|\Pi|}{400\alpha\varepsilon'|\Pi|}\right)^{\mathsf{Err}}\cdot 2^{\mathsf{Err}}\\ &\leq \exp(\Theta(\log(m)\mathsf{Err})). \qquad (2.39)\end{aligned}$$

Again, we use Proposition 2.3.37 to show that the total fraction of random strings that can allow *any* oblivious adversary that makes more than $400\alpha\varepsilon'|\Pi|$ errors to guarantee that $\mathsf{Err} \leq \frac{\varepsilon}{K}\mathsf{CC}$ and that $\mathsf{CC} > 400\alpha|\Pi|m\log(m)$ is at most:

$$\begin{aligned}&\sum_{\mathsf{Err}=400\alpha\varepsilon'|\Pi|}^{C\varepsilon m|\Pi|}\binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}}\cdot 2^{\mathsf{Err}}\cdot\exp(-\Omega(\log(m)\mathsf{Err}/\varepsilon))\\ &\leq \sum_{\mathsf{Err}=400\alpha\varepsilon'|\Pi|}^{C\varepsilon m|\Pi|}\exp(O(\log(m)\mathsf{Err}))\cdot\exp(-\Omega(\log(m)\mathsf{Err}/\varepsilon))\\ &\leq \exp(-\Omega(\log(m)\mathsf{Err}/\varepsilon)) \qquad (2.40)\end{aligned}$$

Finally, putting together Eqs. (2.38) and (2.40) yields the desired result: the fraction of random strings that lead *any* additive oblivious adversary to have too much communication, is low. We conclude that this upper bounds the probability over the randomness of the parties that any non-oblivious party can make the communication too large. □

Now we are ready to finish the proof of Theorem 2.3.33.

Proof of Theorem 2.3.33. We have already showed that with high probability over the randomness of the parties, the communication will be only a constant blowup over $\mathsf{CC}(\Pi)$. Specifically, Corollary 2.3.36 tells us that the communication CC is $O(|\Pi|m\log(m))$ with probability at least $1-\exp(-\Omega(|\Pi|\log(m)))$. For the remainder of this proof, we work under the assumption that CC is bounded by this quantity $\mathsf{CC}_{max} = O(|\Pi|m\log(m))$.

First, note that this means that the adversary cannot corrupt any randomness exchange while having error rate $\frac{\varepsilon}{m\log(m)}$. Indeed, this is because corrupting even a single randomness exchange costs $\Theta(\log(m)|\Pi|)$ errors, and the maximum number of errors that the adversary can commit while staying under the error rate is $O(\varepsilon|\Pi|)$. Hence, we can work under the assumption that all the parties exchange randomness successfully, and hence $J = |E \setminus E'| = 0$.

Now, suppose that the communication complexity is $\mathsf{CC} = O(|\Pi|K)$ and $\mathsf{Err} \leq \frac{\varepsilon}{K}\mathsf{CC} \leq O(\varepsilon|\Pi|)$. We show that the probability of the number of errors being this small yet the number of dangerous rounds D being too large, is exponentially small. Specifically, let k be a sufficiently large constant with respect to C_6 and to the constant factors in the hash collision probability $p = O(1/\mathrm{poly}(m))$. Then for any *oblivious* adversary that makes at most Err errors, we have that

$$\mathbb{P}[(D > k \cdot \mathsf{Err}) \wedge (\mathsf{Err} \leq (\varepsilon/K)\mathsf{CC})] \leq \exp(-\Omega(k \cdot \mathsf{Err} \cdot \log(m)))$$

by Corollary 2.3.26. Specifically, if we let $\mathsf{Err}_{max} = \frac{\varepsilon}{K}\mathsf{CC}_{max}$, then this gives us that the probability that, if the adversary commits Err errors,

$$\begin{aligned}\mathbb{P}[(D > k \cdot \mathsf{Err}_{max}) \wedge (\mathsf{Err} \leq \mathsf{Err}_{max})] &\leq \mathbb{P}[(D > k' \cdot \mathsf{Err}) \wedge (\mathsf{Err} \leq \mathsf{Err}_{max})] \\ &\leq \exp(-\Omega(k \cdot \mathsf{Err}_{max} \cdot \log(m))), \end{aligned} \tag{2.41}$$

by applying Corollary 2.3.26 with Err and k' such that $k' \cdot \mathsf{Err} = k \cdot \mathsf{Err}_{max}$ (and hence $k' \geq k$ is sufficiently large to apply the corollary).

Finally, we union bound Eq. (2.41) over all non-oblivious adversaries that make at most $O(\varepsilon|\Pi|)$ errors to show that, with high probability, no non-oblivious adversary can make D

larger than $k \cdot \mathsf{Err}_{max}$, with high probability. This probability is at most

$$\begin{aligned}
&\sum_{\mathsf{Err}=0}^{\mathsf{Err}_{max}} \binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}} \cdot 2^{\mathsf{Err}} \cdot \mathbb{P}\left[(D > k \cdot \mathsf{Err}_{max}) \wedge \mathsf{Err} \le \mathsf{Err}_{max}\right] \\
&\le \sum_{\mathsf{Err}=0}^{\mathsf{Err}_{max}} \binom{Cm^2\log(m)|\Pi|}{\mathsf{Err}_{max}} \cdot 2^{\mathsf{Err}_{max}} \cdot \mathbb{P}\left[(D > k \cdot \mathsf{Err}_{max}) \wedge \mathsf{Err} \le \mathsf{Err}_{max}\right] \\
&\le \sum_{\mathsf{Err}=0}^{\mathsf{Err}_{max}} \exp(\Theta(\varepsilon|\Pi|\log(m))) \cdot \exp(-\Omega(k \cdot \varepsilon|\Pi| \cdot \log(m))) \\
&\le \exp(-\Omega(k\varepsilon|\Pi| \cdot \log(m))).
\end{aligned}$$

The second line follows from the monotonicity of binomial coefficients with the bottom part much smaller than the top. The third line follows from applying Eq. (1.1) and bringing factors of m and ε into the exponent, and by Eq. (2.41) and plugging in $\mathsf{Err}_{max} = \frac{\varepsilon}{K}\mathsf{CC}_{max} = O(\varepsilon|\Pi|)$. The final line follows from taking the hash sizes (which affect the constants in Eq. (2.41)) and k to be large enough.

We conclude that no non-oblivious adversary can make $D = O(\varepsilon^*|\Pi|)$ while having $\mathsf{Err} \le O(\varepsilon|\Pi|)$ with probability at least $1-\exp(-\Omega(\varepsilon^*|\Pi|\cdot\log(m)))$ over the randomness of the parties, where ε^* is a fixed constant that is sufficiently larger than $10C_6\varepsilon$ and sufficiently smaller than $1/(4200C_7C_6\alpha)$. Combining this with Corollary 2.3.36 and using the fact that D upper bounds the number of hash collisions, we conclude that the non-oblivious adversary cannot make EHC bigger than $O(\varepsilon^*|\Pi|)$ with probability at least $1-\exp(-\Omega(\varepsilon^*|\Pi|\cdot \log(m)))$. Since ε^* is a constant that does not go to 0 with ε, we can absorb it into the Ω. Hence, we get that the non-oblivious adversary cannot make EHC bigger than $O(\varepsilon^*|\Pi|)$ with probability at least $1-\exp(-\Omega(|\Pi| \cdot \log(m)))$.

Finally, recall that Lemma 2.3.10 shows that the potential ϕ rises by at least K in every iteration. This means that after $100|\Pi|$ iterations, the potential is at least $100|\Pi|K$. Combining this with the fact that $\mathsf{EHC} \le O(\varepsilon^*|\Pi|)$, Claim 2.3.32 gives us the desired result. For a fuller treatment of this part of the proof, see the proof of Theorem 2.2.2. □

2.3.5 Conclusion and Future Directions

In this work we considered coding schemes in the multiparty setting that are resilient to insertions, deletions, and substitution noise. These schemes are the first multi-party coding scheme that are computationally efficient despite the presence of *adversarial* noise. The communication blowup incurred by the coding is only a constant, and the resilience we obtain is $O(1/m)$ assuming oblivious adversary or $O(1/m \log m)$ assuming a general adversary.

While we consider error rates of at most $O(1/m)$ in this paper, this is not the optimal noise rate for multiparty interactive coding in general. Indeed, Hoza and Schulman [79] further improve their noise resilience to $\Omega(1/n)$, where n is the number of parties, at the expense of reducing the rate to $1/O(m \log(n)/n)$, which is no longer constant. It is an interesting open question whether there exists a constant-rate coding scheme with noise resilience $\Omega(1/n)$, in any communication model. We note that our resilience level to non-oblivious adversaries stands at $O(1/m \log(m))$; this seems to be inherent to our approach, designed to yield computational efficiency. New ideas might be required in order to improve this resilience.

Even though our noise-resilient protocol increases the communication complexity by only a constant factor, it may blow up the *number of communication rounds* by more than a constant factor. In our model, unlike the fully utilized model, the communication complexity does not determine the round complexity. Specifically, in our model an interactive protocol with communication complexity CC may consist of CC/m rounds (in the case that the network if fully utilized) or may consist of CC rounds (in the case where the communication is very sparse). Simulating either one of these interactive protocols with the algorithms in this paper will take Θ(CC) rounds. It is an interesting question as to whether one can extend our results to simultaneously achieve constant blowup in rounds.

2.4 The Meeting Points Mechanism

In this section we define and analyze the meeting points mechanism [70] and its respective potential $\varphi_{u,v}$. Certain parts of the analysis mostly repeat [70] and are included here for completeness. The main difference from [70] is that the meeting points mechanism is interleaved over several iteration of the simulation protocol. This brings two potential difficulties. First, $\varphi_{u,v}$ may change outside the meeting-point phase. Second, the transcript may change *while the meeting-points mechanism is still in progress*. We eliminate the later by fixing a party's transcript until the meeting points mechanism reports the transcripts of both parties are consistent. The former requires a more careful analysis of $\varphi_{u,v}$, and is handled in Claim 2.4.1.

2.4.1 Meeting Points Protocol Between Two Parties

Below we describe the meeting points protocol that parties do pairwise. We write the algorithm as it is performed by some party u with one of its neighbors v. In all variables below, we will drop the subscript of (u,v) but it is implied. Specifically, k below denotes $k_{u,v}$, and the same is true for E, T, $mpc1$, $mpc2$, and $status$.

Roughly speaking, there are up to two types of actions performed in each round of meeting points. The pair of parties first send each other hashes of their truncated transcripts, and increment their respective E, T_1, or T_2 counters if applicable. In keeping with Haeupler's paper we will call this the *verification* phase of meeting points. Note that the verification phase *always* occurs during a round of meeting point exchange.

After exchanging hashes, the parties judge whether or not to take further action based on the values of E, T_1, and T_2. We will call this the *transition* phase of meeting points. For example, if $2E_{u,v} > k_{u,v}$, then party u will set all its variables in the meeting points computation to 0. We will call these transitions *reset* transitions[16]. Otherwise, if $mpc1_{u,v} > 0.4k_{u,v}$, party u will transition to meeting point 1, and similarly for meeting point 2. These will be called *meeting point* transitions.

[16]This is called an error transition in [70].

Algorithm 11 MeetingPoints(u,v, $S_{i,u,v}$)

```
1: Method called by $u$, with $v \in N(u)$. $S_{i,u,v} \in \{0,1\}^{\Theta(|\Pi| m \log(m))}$ is a large random seed to be split up and used for hashing. $\Pi \leftarrow$ protocol to be simulated.
2: $h \leftarrow$ inner product hash family (Definition 2.1.2) w/ input length[15] $\Theta(|\Pi| m \log(m))$, $p = 2^{-\Theta(\log(m))}$ sufficiently small, $o = \Theta(\log(m))$, $s = |S_{i,u,v}|/10$.
3: $(S_1, S_2, \ldots, S_{10}) \leftarrow S_{i,u,v}$. $S_{i,u,v}$ is split into ten seeds. Wlog we assume that $S_1, \ldots, S_5$ are for the hashes it sends, and $S_6, \ldots, S_{10}$ are for the hashes it uses for comparisons (so $v$ is using $S_6, \ldots S_{10}$ for the hashes it sends, and $S_1, \ldots, S_5$ for comparisons, respectively).
4: $k \leftarrow k + 1$
5: $\widetilde{k} \leftarrow 2^{\lfloor \log k \rfloor}$. Let $c$ be the largest integer such that $c\widetilde{k} \leq |T_{u,v}|$.
6: $T_1 \leftarrow T_{u,v}[1 : c\widetilde{k}], T_2 \leftarrow T_{u,v}[1 : (c-1)\widetilde{k}]$
7: Send $(h_{S_1}(k), h_{S_2}(T_1), h_{S_3}(T_1), h_{S_4}(T_2), h_{S_5}(T_2))$ to neighbor $v$.
8: $(H_k, H^{(1)}_{T_1}, H^{(1)}_{T_2}, H^{(2)}_{T_1}, H^{(2)}_{T_2}) \leftarrow (h_{S_6}(k), h_{S_7}(T_1), h_{S_8}(T_2), h_{S_9}(T_1), h_{S_{10}}(T_2))$
9: Receive $(H'_k, H^{(1)'}_{T_1}, H^{(2)'}_{T_1}, H^{(1)'}_{T_2}, H^{(2)'}_{T_2})$ from our neighbor $v$.
10: if $H_k \neq H'_k$ then
11:     $E \leftarrow E + 1$
12: else if $H^{(1)}_{T_1} = H^{(1)'}_{T_1}$ or $H^{(2)}_{T_1} = H^{(1)'}_{T_2}$ then
13:     $mpc1 \leftarrow mpc1 + 1$
14: else if $H^{(1)}_{T_2} = H^{(2)'}_{T_1}$ or $H^{(2)}_{T_2} = H^{(2)'}_{T_2}$ then
15:     $mpc2 \leftarrow mpc2 + 1$
16: if $k = 1, E = 0$, and $H^{(1)}_{T_1} = H^{(1)'}_{T_1}$ then
17:     $k, mpc1, mpc2 \leftarrow 0$
18:     $status \leftarrow$ "simulate" return $status$
19: if $2E \geq k$ then
20:     $k \leftarrow 0, E \leftarrow 0, mpc1 \leftarrow 0, mpc2 \leftarrow 0$
21:     $status \leftarrow$ "meeting points"
22: else if $k = \widetilde{k}$ then
23:     if $mpc1 > 0.4k$ then
24:         $T_{u,v} \leftarrow T_1$
25:         $k \leftarrow 0, E \leftarrow 0$
26:     else if $mpc2 > 0.4k$ then
27:         $T_{u,v} \leftarrow T_2$
28:         $k \leftarrow 0, E \leftarrow 0$
29:     $mpc1 \leftarrow 0, mpc2 \leftarrow 0$
30:     $status \leftarrow$ "meeting points"
31: else
32:     $status \leftarrow$ "meeting points"
33: return $status$
```

We have the parties use separate seeds for each hash comparison, and so they take the large shared seed between them and split it into many smaller seeds. The reason for this is somewhat technical; it makes the events of hash collisions for the different comparisons independent (when the seeds themselves are independent), which will be useful in our analysis of removing the common random string in the second part of this work [59][17]. It suffices to use a single seed in Meeting Points when the parties share a common random string).

2.4.2 Notation

We establish some notation that will be used in this section.

- $E_{u,v}$ corresponds to the value of E that party u has for the communication link with party v. $E_{u,v}$ is defined similarly.
- $E_{u,v}$,$k_{u,v}$, $mpc1_{u,v}$, and $mpc2_{u,v}$ are all defined as the value of the corresponding variable that party u has for the communication link (u,v). We can also define these with v coming first in the subscript (e.g. $E_{v,u}$); these will correspond to the value that party v has for the link (u,v).
- $WM_{u,v}$ corresponds to the number of wrong matches or mismatches that contribute to the current values of $mpc1_{u,v}$ and $mpc2_{u,v}$ (the counters for party u) on link (u,v). That is, if $(T_1)_{u,v} \in \{(T_1)_{v,u}, (T_2)_{v,u}\}$ but party u does not increment $mpc1$ due to an error, then we increment $WM_{u,v}$ by 1. Similarly, if $(T_1)_{u,v} \notin \{(T_1)_{v,u}, (T_2)_{v,u}\}$ but party u increments $mpc1$ due to an error or hash collision, we increment $WM_{u,v}$ by 1. Define $WM_{v,u}$ similarly, but for the increments and non-increments of party v.
- Let $k^{\{u,v\}} = k_{u,v} + k_{v,u}$. Define $E^{\{u,v\}}, mpc1^{\{u,v\}}, mpc2^{\{u,v\}}$, and $WM^{\{u,v\}}$ similarly.
- Given some number V that depends on the transcript T, define $\Delta(V)$ to be the change

[17]While it helps with our analysis, is not clear that using separate seeds is *necessary* to remove the common random string.

in V that results after one invocation to Meeting Points (Algorithm 11) for all pairs of adjacent parties (i.e. in the Meeting Points phase of Algorithm 1).

- Similarly, given a number V that depends on the transcript T, define $\Delta_{u,v}(V)$ to be the change in V that results in parties u and v running Meeting Points (Algorithm 11) with each other, and no other pair of parties making changes.
- When it is understood that we are only talking about the interaction between a pair of parties u and v, we will drop the superscript $\{u, v\}$ off terms such as $k^{\{u,v\}}, E^{\{u,v\}}$, $WM^{\{u,v\}}$, etc. with the understanding that we are only talking about this pairwise interaction.

2.4.3 Potential Analysis

Following the paper of Haeupler [70], we define $\varphi_{u,v}$ as follows. Let $1 < C_1 < C_2 < C_3 < C_4 < C_5 < C_6 < C_7$, where each C_i is selected to be sufficiently larger than C_{i-1} (or 1 if $i = 1$).

$$\varphi_{u,v} = \begin{cases} C_3 \cdot B_{u,v} - C_2 \cdot k^{\{u,v\}} + C_5 \cdot E^{\{u,v\}} + 2C_6 \cdot WM^{\{u,v\}} & \text{if } k_{u,v} = k_{v,u} \\ C_3 \cdot B_{u,v} + 0.9C_4 \cdot k^{\{u,v\}} - C_4 \cdot E^{\{u,v\}} + C_6 \cdot WM^{\{u,v\}} & \text{if } k_{u,v} \neq k_{v,u} \end{cases} \tag{2.42}$$

Recall that we define our final potential function as follows:

$$\phi = \sum_{(u,v)\in E} (K/m)G_{u,v} - K \cdot \varphi_{u,v} - C_1 KB^* + C_7 K \cdot EHC.$$

We start with some simple claims about $\varphi_{u,v}$ that we use directly in the main proof. The proof of following claim (Claim 2.4.1), unlike the other claims in this section, requires knowledge of the robust protocol (Algorithm 1) beyond the definition of $\varphi_{u,v}$ and the Meeting Points protocol (Algorithm 11).

Claim 2.4.1. *The potential $\varphi_{u,v}$ does not change in the Flag Passing phase. In each of the Rewind and Simulation phases, it changes by at most C_3. In the absence of errors or hash* *collisions between u and v in the* iteration as a whole, *$\varphi_{u,v}$ does not increase in the rewind or simulation phases.*

Proof. The only term in $\varphi_{u,v}$ that changes in these phases is $B_{u,v}$. $B_{u,v}$ does not change in the Flag Passing phase, can change by at most 1 in the Simulation phase, and can change by at most 1 in the Rewind phase. This establishes the first part of the claim.

Now we establish the second part of the claim. First we consider the Simulation phase. At a high level, $B_{u,v}$ can increase in the Simulation phase if one of the following is true: 1) u and v are in disagreement, but both decide to simulate anyway, 2) The adversary puts an error between u and v in the Simulation phase, or 3) u decides not to simulate $\Pi_{u,v}$, but v decides to simulate $\Pi_{v,u}$. We now formalize the three cases and prove that all of them require an error or hash collision between u and v somewhere in the iteration.

In the first case, $\Pi_{u,v} \neq \Pi_{v,u}$ but $netCorrect_u = netCorrect_v = 1$. Note that this implies that $status_{u,v} = status_{v,u} =$ "simulate", since otherwise one of the parties would have $netCorrect = 0$. This means there was an error or hash collision between u and v in the Meeting Points phase to make them think that their transcripts matched. In the second case, there was an error between u and v in the Simulation phase. In the third case, note that v simulates in $\Pi_{v,u}$, so we know that $netCorrect_v = 1$ and v never received $\perp$ from u in the Simulation phase. Furthermore, u does not simulate. This can happen due to one of two reasons. The first case is that $netCorrect_u = 0$, and so u did not want to simulate after the Flag Passing phase. In this case, u would have sent $\perp$ to v. Since v did not receive it, the adversary must have deleted it. The second case is that u had $netCorrect_u = 1$, but received a $\perp$ on the link (u,v). However, $netCorrect_v = 1$, so v would not have sent this $\perp$. Hence, it was inserted by the adversary. Either way, there is an error on the link (u,v) during the Simulation phase.

Now we consider the Rewind phase. If neither u nor v send or receive a rewind message on (u,v), then it is clear that $B_{u,v}$ is unchanged. We assume that any rewind sent on the link (u,v) reaches the recipient. If this is not the case, then there was a deletion on the link (u,v) and we are done. Similarly, we assume that any rewind received on the link (u,v)

was sent by the other party (otherwise it was an insertion). We break the remainder of the proof into cases depending on which parties want to send rewinds on the link (u,v).

Suppose that $status_{u,v} = status_{v,u} =$ "simulate", $alreadyRewound_{u,v} = alreadyRewound_{v,u} = 0$. Then the following actions all happen together in the same round or not at all: u (resp. v) sends "rewind" to v (resp. u), u truncates $\Pi_{u,v}$, v truncates $\Pi_{v,u}$, and $alreadyRewound_{u,v}$ and $alreadyRewound_{v,u}$ are set to 1. If all these actions happen, then $B_{u,v}$ does not increase. This is because $\max\{|\Pi_{u,v}|, |\Pi_{v,u}|\}$ falls by one, and $G_{u,v}$ falls by at most 1. Furthermore, after these actions happen, $alreadyRewound_{u,v} = alreadyRewound_{v,u} = 0$, so u and v will not rewind on (u,v) anymore.

If $status_{u,v} =$ "meeting points" or $alreadyRewound_{u,v} = 1$, then u will not send a rewind message to v nor take any action if it receives a rewind from v. Therefore, if v does not send a rewind message to u, then $B_{u,v}$ is unchanged in this Rewind phase. If v does send a rewind message to u, then we know that just before the message was sent we had $status_{v,u} =$ "simulate" and $alreadyRewound_{v,u} = 0$. If $status_{u,v} =$ "meeting points", then there must have been a hash collision or error between u and v in the Meeting Points phase, otherwise they would both have the same status. Otherwise if $alreadyRewound_{u,v} = 1$, then u has already truncated $\Pi_{u,v}$ by one chunk. So the net change to $B_{u,v}$ from the truncation of $\Pi_{u,v}$ and $\Pi_{v,u}$ that have happened in this rewind phase is nonpositive, as established in the previous paragraph. Furthermore, after this rewind is sent, both parties have $alreadyRewound = 1$, and will not do anything further on this link in the rewind phase. □

Proposition 2.4.2. *At the beginning of any iteration i of the robust protocol (Algorithm 1), the following is true for all pairs $(u,v) \in E$:*

$$0 \le B_{u,v} \le \varphi_{u,v}.$$

As a corollary, $\sum_{(u,v)\in E} \varphi_{u,v} \ge \sum_{(u,v)\in E} B_{u,v}$.

Proof. If $k_{u,v} = k_{v,u}$, follows from fact that we can take $C_3 > 8C_2 + 1$. Thus, if $k_{u,v}$ is large enough such that $C_2 k^{\{u,v\}} > C_3 B_{u,v}$, this means that we have $k_{u,v} = k_{v,u} > 4B_{u,v}$. But we know that when both $k_{u,v}$ and $k_{v,u}$ are larger than $2B_{u,v}$, both parties are including hashes of their pairwise transcript truncated below the chunk $G_{u,v}$, which should match. So this means the fact that $k_{u,v}$ and $k_{v,u}$ increased so much more without the parties doing a meeting point transition means that there were many mismatches due to errors - specifically, $WM^{\{u,v\}} > 0.6(1/2)k^{\{u,v\}}$. Hence, in this case, the sum of all the terms that do not include $B_{u,v}$ is nonnegative.

If $k_{u,v} \neq k_{v,u}$, then this follows from the fact that $k^{\{u,v\}} \geq 2E^{\{u,v\}}$ so $0.9C_4 k^{\{u,v\}} \geq C_4 E^{\{u,v\}}$. □

We would like to lower bound $\Delta(\phi)$ with $\Delta_{u,v}(\phi)$. To this end, we define a lower bound on $\Delta_{u,v}(\phi)$ as follows:

$$\widetilde{\Delta_{u,v}(\phi)} = (K/m)\Delta_{u,v}(G_{u,v}) - K \cdot \Delta_{u,v}(\varphi_{u,v}) + C_1 K \cdot \Delta_{u,v}(G^*).$$

Claim 2.4.3. $\Delta(\phi) \geq \sum_{(u,v)\in E}\left(\widetilde{\Delta_{u,v}(\phi)} + C_7 K \Delta_{u,v}(EHC)\right)$.

Proof.

$$\begin{aligned}
\Delta(\phi) &= (K/m)\Delta(\sum G_{u,v}) - K \cdot \Delta(\sum \varphi_{u,v}) - C_1 K \cdot \Delta(B^*) + C_7 K \cdot \sum \Delta_{u,v}(EHC) \\
&\geq \sum (K/m)\Delta(G_{u,v}) - K \cdot \sum \Delta(\varphi_{u,v}) + C_1 K \cdot \Delta(G^*) + C_7 K \cdot \sum \Delta_{u,v}(EHC) \\
&= \sum (K/m)\Delta(G_{u,v}) - K \cdot \sum \Delta(\varphi_{u,v}) + C_1 K \cdot \min_{(u,v)\in E}(\Delta_{u,v}(G^*)) + C_7 K \cdot \sum \Delta_{u,v}(EHC) \\
&\geq \sum (K/m)\Delta(G_{u,v}) - K \cdot \sum \Delta(\varphi_{u,v}) + C_1 K \cdot \sum \Delta_{u,v}(G^*) + C_7 K \cdot \sum \Delta_{u,v}(EHC) \\
&= \sum_{(u,v)\in E} \left(\widetilde{\Delta_{u,v}(\phi)} + C_7 K \Delta_{u,v}(EHC)\right).
\end{aligned}$$

The second line follows from the fact that parties do not simulate in the meeting points phase, so $\Delta(B^*) = \Delta(H^*) - \Delta(G^*) \leq -\Delta(G^*)$. The third line follows from the fact that the parties all do meeting points in parallel and the definition of G^*: after doing the meeting points phase, there is some pair of parties (u,v) such that the maximum chunk number that they have simulated correctly is the new value of G^*. Then, by definition, we have that $\Delta_{u,v}(G^*) = \Delta(G^*)$. The fourth line follows from the fact that parties do not simulate in meeting points, so $\Delta_{u,v}(G^*) \leq 0$. □

The main claim that we establish in this section is that if $status_{u,v}$ or $status_{v,u}$ is "meeting points" after the Meeting Points algorithm, then the function $\widetilde{\Delta_{u,v}(\phi)}$ rises by $\Omega(K)$ in the meeting points phase between parties u and v in the absence of errors and hash collisions. Furthermore, if $status_{u,v}$ = "simulate" after Meeting Points, then $\varphi_{u,v}$ does not change. Note that this implies that the potential ϕ rises by at least $\Omega(c \cdot K)$ where c is the number of pairs of adjacent parties (u,v) such that $status_{u,v}$ = "meeting points" and no errors or hash collisions occur between them, by Claim 2.4.3. The proof of this is very similar to the main proof in [70], where he effectively shows that $G_{u,v} - \varphi_{u,v}$ rises by $\Omega(1)$ in each iteration of Meeting Points and Simulation.

In the rest of the section, we will fix parties u and v and focus on how the potential between them changes after they do Meeting Points (Algorithm 11) with each other. As noted earlier, we will drop the superscript $\{u,v\}$ off terms such as $k^{\{u,v\}}, E^{\{u,v\}}$, $WM^{\{u,v\}}$, etc. with the understanding that we are only talking about this pairwise interaction.

Proposition 2.4.4. *Fix parties u and v such that $(u,v) \in E$. If $status_{u,v} = status_{v,u}$ = "simulate" after Meeting Points, then $\varphi_{u,v}$ is unchanged after Meeting Points, and $\widetilde{\Delta_{u,v}(\phi)} = 0$.*

Proof. If we have $status_{u,v} = status_{v,u}$ = "simulate" after Meeting Points, then all variables in $\varphi_{u,v}$ remain unchanged when the Meeting Points method returns. We note that no kind of truncation occurs in this case, so neither $G_{u,v}$ nor G^* change after the Meeting Points phase in this case, hence $\widetilde{\Delta_{u,v}(\phi)} = 0$ □

Lemma 2.4.5. *Fix parties u and v such that $(u,v) \in E$. Then the* verification *phase of Meeting Points between u and v causes the potential $\varphi_{u,v}$ to rise by at most $5C_6$, and we have that $\widetilde{\Delta_{u,v}(\phi)} \geq -5C_6K$ when only taking into account changes from the verification phase. Furthermore, in the absence of errors or hash collisions, $\varphi_{u,v}$ falls by at least five, and $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$ when only taking into account changes from the verification phase.*

Proof. We start by noting that E and WM increment by at most 2 in any verification phase. Furthermore, both $k_{u,v}$ and $k_{v,u}$ are incremented by one: this means that if they start equal, they will stay equal. Conversely, if they start different, they will stay different. Furthermore, note that, since this is a verification phase, no transition occurs, so $B_{i,i+1}$ remains the same. Thus $\varphi_{u,v}$ rises by at most $5C_6$, if we take C_6 to be sufficiently large with respect to C_5.

Case $k_{u,v} = k_{v,u}$:

WM increments only if there was an error or hash collision in the verification phase. Now suppose there is no error or hash collision. In this case, E and WM do not increment, and k increments by 2 (one increment for each party), so $\varphi_{u,v}$ falls by at least five by taking $C_2 > 2.5$.

Case $k_{u,v} \neq k_{v,u}$:

Note that WM, by definition, increments only in the presence of an error, and it will increase by at most 2. This contributes $4C_6$ to $\varphi_{u,v}$. Suppose that E does not increment by 2. This means that there was a hash collision or error - otherwise, both parties would have incremented E. Now suppose that E does increment by 2. In this case, $\varphi_{u,v}$ falls by at least $(-0.9C_4 + C_4)2$. By choosing C_4 to be a sufficiently large number, this is at least five.

The conclusions about $\widetilde{\Delta_{u,v}(\phi)}$ follow from the fact that $G_{u,v}$ and G^* remain unchanged in the verification phase, so $\widetilde{\Delta_{u,v}(\phi)} = -K\varphi_{u,v}$. □

Before proceeding with the proof, we define some notation we will use.

Notation for remainder of section:

- In proofs, we will often drop superscripts on quantities such as $k^{\{u,v\}}, WM^{\{u,v\}}, E^{\{u,v\}}$, with the understanding that all quantities have $\{u,v\}$ as an implied superscript.

- Quantities like $k_{u,v}, WM_{u,v}, E_{u,v}$ will refer to the value of these variables just before the transition phase (that is, *after* the verification phase). We note that there is one proposition for which we will use $k_{u,v}$ to denote the value of the variable before the previous verification phase as well - we will be explicit about this abuse of notation in this case.

- Define $k'_{u,v}, WM'_{u,v}, E'_{u,v}$ to be the value of the corresponding variables after the transition phase. Define $k'_{v,u}, WM'_{v,u}, E'_{v,u}$ similarly. Finally, define $k' = k'_{u,v} + k'_{v,u}$, and define E' and WM' similarly.

- We abuse (our own) previous notation and define $\Delta(k_{u,v})$ to be $k'_{u,v} - k_{u,v}$ - that is, the difference in the variable after transitioning. Define $\Delta(E_{u,v})$, $\Delta(WM_{u,v})$, $\Delta_{u,v}(G^*)$, and $\Delta(B_{u,v})$ similarly as the change incurred in the relevant value by the transition phase between u and v.

Lemma 2.4.6. *Fix parties u and v such that $(u,v) \in E$, and suppose that $status_{u,v}$ = "meeting points" after the Meeting-Points phase. Then Meeting-Points phase between u and v causes the potential $\varphi_{u,v}$ to increase by at most $5C_6$. In the absence of errors or hash collisions, $\varphi_{u,v}$ decreases by at least five. Furthermore, we have that $\widetilde{\Delta_{u,v}(\phi)} \geq -5C_6 \cdot K$, and in the absence of errors or hash collisions, $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$.*

Note that the above lemma includes the verifiation and transition phases. We have already established that this is true for the verification phase *alone* in Lemma 2.4.5. If we could also establish that $\varphi_{u,v}$ does not increase and $\widetilde{\Delta_{u,v}(\phi)}$ does not decrease in the transition phase, we would be able to conclude this lemma as a corollary. However, there is one case for which we cannot do this - in this case, we need to lump together the transition phase previous verification phase to argue that the potential rises enough there to pay for a possible decrease in our transition phase. This will occur in Proposition 2.4.7. We now split Lemma 2.4.6 into cases and prove each case separately. We assume that there is no error in the transition phase. Since the decisions of the parties here only depend on their state, any error does not affect the transition and its corresponding affect on $\varphi_{u,v}$ or $\widetilde{\Delta_{u,v}(\phi)}$.

Proposition 2.4.7. *Fix parties u and v such that $(u,v) \in E$. Suppose that $k_{u,v} \neq k_{v,u}$, and exactly one party does a meeting point or reset transition. Then the current invocation of Meeting Points* as a whole *causes $\varphi_{u,v}$ to rise by at most $5C_6$ and $\widetilde{\Delta_{u,v}(\phi)} \geq -5C_6 \cdot K$. If no error or hash collision occurs, then $\varphi_{u,v}$ falls by at least five, and $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$.*

Proof of Proposition 2.4.7. Suppose wlog that the transitioning party is party u. Since party v did *not* transition, we have that $k'_{u,v} \neq k'_{v,u}$ after the transition as well. We now analyze the parameters for party u, to analyze how party u's contribution to the potential changes. We assume that $WM_{u,v}$ was initially 0, otherwise the decrease in $WM_{u,v}$ to 0 will increase $\varphi_{u,v}$. Now, we know that party u will set $k_{u,v}$ and $E_{u,v}$ to 0. Setting $k_{u,v}$ to 0 will result in an increase in potential, and setting $E_{u,v}$ to 0 will result in a decrease in potential. The net change of $\varphi_{u,v}$ from these two actions is $-0.9C_4k_{u,v} + C_4E_{u,v}$.

Note that for a meeting point transition, we have that $E_{u,v} < 0.5k_{u,v}$, so the expression above is $-0.9C_4k_{u,v} + C_4E_{u,v} \geq -0.4C_4k_{u,v}$. Note that a meeting points transition can affect $B_{u,v}$, G^*, and $G_{u,v}$ as well. However, that the change in each of these values is at most $2k_{u,v}$, and the constants multiplying them in $\widetilde{\Delta_{u,v}(\phi)}$ are C_3, C_1, and 1 respectively. Since we can take these to be much smaller than C_4, the affect on $\Delta(\varphi_{u,v})$ (resp. $\widetilde{\Delta_{u,v}(\phi)}$) from changes in these variables are negligible compared to $-0.4C_4k_{u,v}$ (resp. $0.4C_4k_{u,v} \cdot K$). Hence, by taking C_4 to be large enough, we get the desired result, that $\varphi_{u,v}$ falls by at least five and $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$.

Now we turn our attention to reset transitions. Note that for reset transitions, $G_{u,v}$ and G^* are unchanged, so $\widetilde{\Delta_{u,v}(\phi)} = -K\Delta(\varphi_{u,v})$. Recall that, when $2E_{u,v} \geq k_{u,v}$, u will reset. So, we know that *before* her most recent increment of $k_{u,v}$ in the last verification phase (and possibly $E_{u,v}$), we either had that $k_{u,v}, E_{u,v} = 0$ or $2E_{u,v}$ is strictly less than $k_{u,v}$. This means that, after potentially incrementing $E_{u,v}$ and $k_{u,v}$ in the following iteration, we have that (currently) $E_{u,v} \leq 0.5k_{u,v} + 0.5$.[18]. Plugging into the expression for potential difference above, we see that

$$\Delta(\varphi_{u,v}) \leq C_4(-0.9k_{u,v} + E_{u,v}) \leq C_4(-0.4k_{u,v} + 0.5). \tag{2.43}$$

Note that when $k_{u,v} > 1$, this is at most -5 for sufficiently large C_4, and hence $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$. Combining this with the rise in potential during the verification phase (Lemma 2.4.5), we conclude the proof of Proposition 2.4.7 in the case when $k_{u,v} > 1$.

[18]Note that this inequality also holds if we started with $k_{u,v}$, $E_{u,v}$ equal to 0.

When $k_{u,v} = 1$, we note that, before the previous verification phase, we must have had $k_{u,v} = 0$. So if we compare $k_{u,v}$ *before the verification phase* to $k'_{u,v}$ *after the transition phase*, we see that they are both equal. The same is true of $E_{u,v}$ before the verification phase and $E'_{u,v}$. So party u's contribution to $\varphi_{u,v}$ does not increase after one iteration of the meeting points protocol, and all the other terms in $\widetilde{\Delta_{u,v}(\phi)}$ are also unchanged. But what about party v? By assumption, party v does not transition. Further, in the absence of errors and hash collisions we know that $status_{v,u}$ = "meeting points" after Meeting Points, since $k_{u,v} \neq k_{v,u}$. Therefore, v goes through a verification phase. By Lemma 2.4.5 and the fact that v does not transition, we get that the contribution of party v to $\varphi_{u,v}$ decreases by at least five in the absence of errors and hash collisions, and increases by at most $5C_6$ in their presence. Hence, overall, $\varphi_{u,v}$ falls by at least five in the absence of errors, and rises by at most $5C_6$ in the presence of errors. □

Note on notation: $\widetilde{\Delta_{u,v}(\phi)}$ changes meaning for the rest of the proof, to match with $\Delta(\cdot)$.

Proposition 2.4.7 was the only case where we needed to lump together verification and transition phases to argue that the potentials behave like we want. For the remainder of the argument, it suffices to show that $\varphi_{u,v}$ falls in the transition phase, and that $\widetilde{\Delta_{u,v}(\phi)} > 0$ rises. Hence, we abuse our previous notation to define $\widetilde{\Delta_{u,v}(\phi)} := (K/m)\Delta_{u,v}(G_{u,v}) - K \cdot \Delta(\varphi_{u,v}) + C_1 K \cdot \Delta_{u,v}(G^*)$, where we recall that $\Delta_{u,v}(\cdot)$ is now defined to be the change in a variable after the transition phase *only*.

Proposition 2.4.8. *Fix parties u and v such that $(u,v) \in E$. Suppose that $k_{u,v} \neq k_{v,u}$, and both parties do some transition. Then $\varphi_{u,v}$ falls by at least one in the transition phase. Furthermore, $\widetilde{\Delta_{u,v}(\phi)} \geq K$.*

Proof of 2.4.8. Since both parties transition, we will have $k'_{u,v} = k'_{v,u} = 0$ after the transition. Furthermore, we will have $E'_{u,v} = E'_{v,u} = WM'_{u,v} = WM'_{v,u} = 0$ due to the transitioning. Hence, $\Delta(\varphi_{u,v}) \geq -0.9C_4 k + C_4 E - C_6 WM$. Just like in the proof of Proposition 2.4.7, we note that the reset condition (Line 19) implies that we have $E_{u,v} \leq 0.5k_{u,v} + 0.5$, and similarly $E_{v,u} \leq 0.5k_{v,u} + 0.5$. So $E \leq 0.5k + 1$, and so we get that

$$\Delta(\varphi_{u,v}) \leq C_4(-0.4k + 1).$$

Since we know that both $k_{u,v}$ and $k_{v,u}$ are greater than 1 after the verification phase and we also know that $k_{u,v} \neq k_{v,u}$, we conclude that $k_{u,v} + k_{v,u} \geq 3$. Therefore, the above expression is at most $-0.2C_4$. This establishes that $\Delta(\varphi_{u,v}) \leq -1$. To see that indeed $\widetilde{\Delta_{u,v}(\phi)} \geq K$, we additionally note that G^* and $G_{u,v}$ change by at most $2k$, and so we get that $\widetilde{\Delta_{u,v}(\phi)} \geq K \cdot (-2k + 0.4C_4k - 2C_1k - C_4) \geq K \cdot k(0.4C_4 - 2C_1 - 2) - C_4$. By taking C_4 sufficiently large so that $0.05C_4 \geq 2C_1 + 2$ and using the fact that $k \geq 3$, we see that this in turn is $\geq 0.05C_4K$, which is greater than K since we needed $C_4 > 20$ earlier. □

Proposition 2.4.9. *Fix parties u and v such that $(u,v) \in E$. Suppose that $k_{u,v} = k_{v,u}$, and exactly one party transitions. Then $\varphi_{u,v}$ falls by at least one in the transition phase. Furthermore, $\widetilde{\Delta_{u,v}(\phi)} \geq K$.*

Proof of Proposition 2.4.9. Note that since only one party transitions, $k'_{u,v} \neq k'_{v,u}$. Wlog, suppose that the transitioning party is u. We will directly show that $\widetilde{\Delta_{u,v}(\phi)} \geq K$: since the other quantities in the definition of $\widetilde{\Delta_{u,v}(\phi)}$ do not increase in the Meeting Points phase, this implies that $\Delta_{u,v}(\varphi_{u,v}) \leq -1$.

Let us suppose that the transition was a reset transition, so $E_{u,v} \geq 0.5k_{u,v}$. Then u's contribution to the potential will rise, since it sets $E'_{u,v} = k'_{u,v} = 0$ and we can take $C_5 > 2C_2 + 1$. But a priori it seems possible that party v may have its contribution to $\varphi_{u,v}$ *fall*. This is because the contribution of $k_{v,u}$ to the potential was $-C_2k_{v,u}$, but after party u transitions the contribution is $0.9C_4k'_{v,u}$, since after u's transition we are using the potential function for unequal k's. To address this, we note that $k'_{v,u} = k_{v,u} = k_{u,v} \leq 2E_{u,v}$, where the first equality follows because party v did not transition and the second equality holds by assumption. Hence, the change in $\varphi_{u,v}$ is at least $-C_5E_{u,v} + C_2k_{u,v} + C_2k_{v,u} + C_4k'_{v,u} \geq (-0.5C_5 + 2C_2 + C_4)k_{u,v}$. For sufficiently large choice of C_5, this quantity is at most -1. Since $G_{u,v}$ and G^* do not change for a reset trasition, this also implies that $\widetilde{\Delta_{u,v}(\phi)} \geq K$.

Now let us suppose that the transition was a meeting point transition. For simplicity, assume that party u transitions to meeting point 1; identical reasoning will hold for transitioning to meeting point 2.

Note that u only transitions when $mpc1_{u,v} \geq 0.4k_{u,v}$. Since v is not transitioning, we know that $mpc1_{v,u} < 0.4k_{v,u} = 0.4k_{u,v}$ and $mpc2_{v,u} < 0.4k_{u,v}$. Furthermore, we know that for the last $0.5k_{u,v}$ iterations (if $k_{u,v} = 1$, then for 1 iteration), both parties have exchanged hashes of the same meeting points. Either there was truly a match among these meeting points or there was not. If there was truly a match (wlog say it is with v's first meeting point), then we know that $WM \geq WM_{v,u} > 0.1k_{v,u} = 0.05k$, since v did not increment either $mpc1$ more than $0.4k_{u,v}$ times. Since $k'_{u,v} \neq k'_{v,u}$ after the transition, the WM term in $\varphi_{u,v}$ falls by $0.05C_6 \cdot k$ as a result. If there was not truly a match, then we know that $WM_{u,v} \geq 0.4k_{u,v}$, since u incremented its $mpc1$ counter $0.4k_{u,v}$ times despite the lack of a true match. $WM_{u,v}$ resets to 0 after u transitions, so the of WM term in $\varphi_{u,v}$ falls by at least $0.2C_6 \cdot k$ after the transition.

Furthermore, the contribution of each of the other terms in $\varphi_{u,v}$ is at most $2C_5 \cdot k$ due to this transition. Hence, by taking C_6 to be sufficiently large with respect to C_5, we get that $\varphi_{u,v}$ rises by at least $\Omega(C_6 \cdot k)$, which is at least 1 for sufficiently large C_6. Furthermore, $G_{u,v}$ and G^* also only change by at most $2k$, and so by taking C_6 to be sufficiently large, we get that $\widetilde{\Delta_{u,v}(\phi)} \geq K$. □

Proposition 2.4.10. *Fix parties u and v such that $(u,v) \in E$. Suppose that $k_{u,v} = k_{v,u}$, and both parties transition. Then $\varphi_{u,v}$ falls by at least one in the transition phase. Furthermore, $\widetilde{\Delta_{u,v}(\phi)} \geq K$.*

Proof of Proposition 2.4.10. The main difference from Proposition 2.4.9 is that we have $k'_{u,v} = k'_{v,u}$ after the transition. Note that $|\Delta(B_{u,v})|, |\Delta(G_{u,v})|$, and $|\Delta_{u,v}(G^*)|$ are all upper bounded by $2k$. Suppose that at least one party (wlog, u) does a reset transition. Note that this transition can only decrease WM, which in turn only decreases $\varphi_{u,v}$, so we assume that $WM = 0$. Furthermore, recall that $E_{u,v} \geq k_{u,v}/2 = k/4$. Then the difference in $\widetilde{\Delta_{u,v}(\phi)}$

caused after both parties transition is at least

$$\begin{aligned}\widetilde{\Delta_{u,v}(\phi)} &\geq K(\Delta(G_{u,v}) + C_5 E_{u,v} + C_2\Delta(k) - C_3\Delta(B_{u,v}) + C_1\Delta_{u,v}(G^*)) \\ &\geq ((C_5/4)k - 2C_3k - C_2k - 2C_1k - 2k)K \geq ((C_5/4) - 7C_3)k \cdot K\end{aligned}$$

By taking C_5 to be larger than $28C_3 + 1$ and noting that $k = k_{u,v} + k_{v,u} > 1$, we see that $\widetilde{\Delta_{u,v}(\phi)} > K$.

Now suppose both parties do a meeting point transition. Suppose that $k_{u,v} > 4B_{u,v}$. Note that $k_{u,v}$ is a power of two by definition since a meeting point transition is occurring. So if $B_{u,v} > 0$, then $k_{u,v}$ is divisible by 4. Then we must have had at least $k_{u,v}/4$ iterations where u had some value for $k_{u,v}$ that was at least $k_{u,v}/4 > B_{u,v}$, and the same for v with $k_{v,u}$. In this case, one of party u's two meeting points corresponds to $\Pi_{u,v}[1 : c(k_{u,v}/4)]$, where c is defined to be the largest integer such that $c \cdot k_{u,v}/4 \leq G_{u,v}$. This uses the fact that $k_{u,v} > 0$ after the Meeting Points phase implies that $status_{u,v}$ = "meeting points", and that this prevents u from simulating or rewinding $\Pi_{u,v}$. The analogous fact is true of v as well, for identical reasons. Then, by the definition of $G_{u,v}$, we have that

$$\Pi_{u,v}[1 : c(k_{u,v}/4)] = \Pi_{v,u}[1 : c(k_{u,v}/4)].$$

However, the parties did not transition at step $k_{u,v}/2$ – this means that we must have $WM \geq 0.4(k_{u,v}/2) = 0.1k$. Since C_6 is sufficiently large, we get that $\varphi_{u,v}$ falls by at least 1 and that $\widetilde{\Delta_{u,v}(\phi)} \geq K$.

Note that if $B_{u,v} = 0$, then it is possible that we have $k_{u,v} \in \{1, 2\}$, which we now address for completeness. If $k_{u,v} = 1$, then the parties exchanged a single meeting point and are now doing a meeting point transition. But it cannot be that their hashes matched for the first meeting point - if this were the case, they would have made their status "simulate" instead of going into the transition phase. But since we know $B_{u,v} = 0$, their hashes should have matched on the first meeting point. Hence, we get that $WM \geq k_{u,v} = 1$. If $k_{u,v} = 2$, then note that the parties did not match their Meeting Points when they had previously had $k_{u,v} = k_{v,u} = 1$. Since $B_{u,v} = 0$, they should have matched, and so we get that $WM \geq 0.5k_{u,v}$. Hence, either way, WM is sufficiently large so that $\varphi_{u,v}$ falls by at least 1 and that $\widetilde{\Delta_{u,v}(\phi)} \geq K$.

Now we assume that $k_{u,v}, k_{v,u} \leq 4B_{u,v}$. Note that this implies that $B_{u,v} > 0$, since $k_{u,v}$ and $k_{v,u}$ are both at least 1. First, we consider the case where $B'_{u,v} \neq 0$. In this case, their communication before now must have had at least $0.4k_{u,v}$ hash collisions or corruptions to make them both increment their meeting point counters enough to transition, and so WM is at least $0.4k_{u,v}$ before the transition. The decrease in WM from the transition means that $\varphi_{u,v}$ falls by at least 1 and that $\widetilde{\Delta_{u,v}(\phi)} \geq K$.

Now assume that $B'_{u,v} = 0$. Then the potential change from this meeting point transition is at least

$$K \cdot (\Delta(G_{u,v}) + C_1\Delta(G^*) + C_2\Delta(k) - C_3\Delta(B_{u,v})) \geq K \cdot ((1 + C_1 + C_2)(-4k) - C_3(-k/8)).$$

By taking C_3 to be large enough, we get that $\widetilde{\Delta_{u,v}(\phi)} \geq K$.

Since $\widetilde{\Delta_{u,v}(\phi)} > K$, $\varphi_{u,v}$ falls by at least one, as $\Delta_{u,v}(G^*)$ and $\Delta(G_{u,v})$ are both nonpositive. □

Proof of Lemma 2.4.6. First, note that an argument basically identical to the proof of Proposition 2.4.4 shows that any party v with $status_{v,u}$ = "simulate" after Meeting Points does not contribute any change to $\varphi_{u,v}$. Hence, we can ignore these parties when establishing Lemma 2.4.6.

- Suppose neither party transitions. Then the only change to $\varphi_{u,v}$ is in the verification phase, and Lemma 2.4.5 establishes the claim.

- Suppose one party transitions and $k_{u,v} \neq k_{v,u}$. Then Proposition 2.4.7 establishes the claim.

- Suppose both parties transition and $k_{u,v} \neq k_{v,u}$. Then Lemma 2.4.5 and Proposition 2.4.8 establish the claim.

- Suppose one party transitions and $k_{u,v} = k_{v,u}$. Then Lemma 2.4.5 and Proposition 2.4.9 establish the claim.
- Suppose both parties transition and $k_{u,v} = k_{v,u}$. Then Lemma 2.4.5 and Proposition 2.4.10 establish the claim.

□

Now we can prove the final lemma of this section, will be used directly in the proof that the potential ϕ rises during the Meeting Points phase (Lemma 2.2.3).

Lemma 2.4.11. *Let c be the number of pairs $(u,v) \in E$ such that $status_{u,v}$ or $status_{v,u}$ is "meeting points" after the Meeting Points phase. Let ℓ_1 denote the number of errors and hash collisions that occur in the network during the Meeting Points phase. Then after all adjacent parties do Meeting Points, the overall potential rise $\Delta(\phi)$ is at least $5c \cdot K + 0.4C_7\ell_1 \cdot K$.*

Now we can prove Lemma 2.4.11

Proof of Lemma 2.4.11. By Claim 2.4.3, we recall that

$$\Delta(\phi) \geq \sum_{(u,v)\in E} \left(\widetilde{\Delta_{u,v}(\phi)} + C_7 K \Delta_{u,v}(EHC) \right)$$

For any pair (u,v) that have an error or hash collision between them during the Meeting Points phase, $\Delta_{u,v}(EHC) \geq 1$, so we get that $\widetilde{\Delta_{u,v}(\phi)} + C_7 K \Delta_{u,v}(EHC) \geq C_7 \cdot K - 5C_6 \cdot K \geq 0.5C_7 \cdot K$, where this follows from Lemma 2.4.6 in the case where either $status_{u,v}$ or $status_{v,u}$ was "meeting points". In the case where both parties have $status_{u,v}$ = "simulate,", Proposition 2.4.4 gives us that $\varphi_{u,v}$ is unchanged after Meeting Points, and so are G^* and $G_{u,v}$. Hence, $\widetilde{\Delta_{u,v}(\phi)} \geq C_7 \cdot K$.

For any pair (u,v) such that $status_{u,v}$ = " meeting points" and there is no error between them, Lemma 2.4.6 gives us that $\widetilde{\Delta_{u,v}(\phi)} \geq 5K$.

For any pair (u,v) such that $status_{u,v} = status_{v,u}$ = "simulate" after Meeting Points and there is no error between them, Proposition 2.4.4 gives us that $\widetilde{\Delta_{u,v}(\phi)} \geq 0$.

Combining these facts, we get that

$$\begin{aligned}\Delta(\phi) &\geq \sum_{(u,v)\in E} \left(\widetilde{\Delta_{u,v}(\phi})+C_7K\Delta_{u,v}(EHC)\right)\\ &\geq 5c\cdot K+0.4C_7\ell_1\cdot K)\end{aligned}$$

where we take C_7 to be sufficiently large such that $C_7-5C_6\geq 0.5C_7\geq 0.4C_7+5$ $\square$

2.5 Improving the Noise Rate with a CRS

2.5.1 Overview of Scheme

In this section we detail a coding scheme that tolerates fully general (non-oblivious) adversaries with error rate $1/m \log\log(m)$. This improves the noise rate achieved in Section 2.3, in the setting where the parties can share a common random string. The parties use a common hash function h_1 which they agree on via their common random string, in the exact way that they do for the scheme for oblivious adversaries in Section 2.2.

In order to make this scheme robust to *non-oblivious* adversaries that get to see the common random string, they need to increase the output size of the shared hash to $\Theta(\log m)$, just like in Section 2.3. In addition, each party generates *fresh* randomness in each iteration for each of its links, and shares these random strings with their neighbors. The parties use this randomness to further hash the outputs of the common hash function h_1 down to a constant size.

Since the randomness used for this second hash is generated fresh in each iteration, the adversary cannot design their errors in previous iterations to cause hash collisions in this second hash, which allows the output size of this hash to be $\Theta(1)$. This technique is also used in the two-party setting by Haeupler [70] to improve the rate of the coding scheme.

2.5.2 Protocol

Algorithm 12 RobustProtocolV2 (for party u) for non-oblivious adversaries

```
1: InitializeState()
2: Let $K := m \log\log(m)$ (so chunk size is $5m \log\log(m)$).
3: $S = \{S_{i,u,v}\}_{i,u,v} \leftarrow \left(\{0,1\}^{\Theta(|\Pi|K)}\right)^{100|\Pi||E|}$ uniform shared randomness.
4: for $i = 1$ to $100|\Pi|$ do
5:     for all $v \in N(u)$ in parallel do                          ▷ meeting points
6:         $status_{u,v} \leftarrow MeetingPointsV2(u, v, S_{i,u,v})$
7:     Rest same as in Algorithm 1.
```

Theorem 2.5.1. *Assume a network $G = (V, E)$ with $n = |V|$ parties and $m = |E|$ links. Suppose Π is a multiparty protocol over the network G with communication complexity $\mathsf{CC}(\Pi)$, binary alphabet and fixed order of speaking. Let $|\Pi| = \frac{\mathsf{CC}(\Pi)}{5m\log\log(m)}$ and let $\varepsilon > 0$ be a sufficiently small constant. Algorithm C correctly simulates Π with communication complexity $O(\mathsf{CC}(\Pi))$ with probability at least $1 - \exp(-\Omega(|\Pi|))$ over the randomness of the parties, in the presence of a non-oblivious adversary limited to a noise fraction at most $\varepsilon/m\log\log(m)$, as long as the parties are allowed to share a CRS (common random string).*

2.5.3 Bounding Hash Collisions for Non-oblivious Adversaries assuming a CRS

We denote the collision probabilities of the hash functions h_1 and h_2 by p_1 and p_2 respectively, where we note that $p_1 = \frac{1}{m^{\Theta(1)}}$ and p_2 is a sufficiently small constant. Similar to the analysis in Section 2.2.4, we create a process $\psi_{u,v}$ that upper bounds $\varphi_{u,v}$, but now we take into account hash collisions from both hash functions. We will use the terminology that a hash function h collides in (i, u, v) if there is a hash collision when u and v exchange hashes in iteration i.

Similar to the definition in Section 2.2.4, define $X_{i,u,v}$ for iterations when $\varphi_{u,v} > 0$ as

$$X_{i,u,v} = \begin{cases} 1 & \text{if } h_1 \text{ collides in } (i,u,v) \\ 0 & \text{otherwise} \end{cases}$$

and define $Y_{i,u,v}$ for iterations when $\varphi_{u,v} > 0$ as

$$Y_{i,u,v} = \begin{cases} 1 & \text{if } X_{i,u,v} = 0 \text{ and } h_2 \text{ collides in } (i,u,v) \\ 0 & \text{otherwise} \end{cases}$$

Then we define $\psi_{u,v}$ as follows: We argue that $\psi_{u,v}$ is an upper bound on $\varphi_{u,v}$ (that is, Lemma 2.2.15 holds with this definition of $\psi_{u,v}$). Indeed, in the case of a hash collision in

Algorithm 13 MeetingPointsV2(u,v, $S_{i,u,v}$) for non-oblivious adversaries with CRS

```
1: Method called by $u$, with $v \in N(u)$, $S_{i,u,v} \in \{0,1\}^{\Theta(|\Pi|K)}$
2: $h_1 \leftarrow$ inner product hash function (Definition 2.1.2) with input length $\Theta(|\Pi|K)$, $p_1 = \frac{1}{m^{\Theta(1)}}$ for a
   sufficiently small exponent, $o_1 = \Theta(\log(m))$, $s = |S_{i,u,v}|$.
3: $h_2 \leftarrow$ binary hash family from Corollary 2.3.8 with input length $o_1$, $p_2 = \Theta(1)$ sufficiently small,
   $o_2 = \Theta_1$, and $s_2 = \Theta(\log\log(m))$.
4: $S_1 \leftarrow S_{i,u,v}$
5: $S_2 \overset{\text{unif}}{\leftarrow} \{0,1\}^{s_2}$ are fresh privately generated bits.
6: $h_S \leftarrow h_{2,S_2}(h_{1,S_1}(\cdot))$
7: $k \leftarrow k+1$
8: $\widetilde{k} \leftarrow 2^{\lfloor \log k \rfloor}$. Let $c$ be the largest integer such that $c\widetilde{k} \leq |T_{u,v}|$.
9: $T_1 \leftarrow T_{u,v}[1 : c\widetilde{k}], T_2 \leftarrow T_{u,v}[1 : (c-1)\widetilde{k}]$
10: Send $(S_2, h_S(k), h_S(T_1), h_S(T_2))$ to neighbor $v$.
11: Receive $(S_2', H_k', H_{T_1}', H_{T_2}')$ from our neighbor $v$.
12: $h_S' \leftarrow h_{2,S_2'}(h_{1,S_1}(\cdot))$
13: $(H_k, H_{T_1}, H_{T_2}) \leftarrow (h_S'(k), h_S'(T_1), h_S'(T_2))$.
14: **if** $H_k \neq H_k'$ **then**
15:     $E \leftarrow E+1$
16: **if** $k = 1, E = 0$, and $H_{T_1} = H_{T_1}'$ **then**
17:     $k \leftarrow 0$
18:     $status \leftarrow$ "simulate" **return** $status$
19: **if** $H_{T_1} = H_{T_1}'$ or $H_{T_1} H_{T_2}'$ **then**
20:     $mpc1 \leftarrow mpc1 + 1$
21: **else if** $H_{T_2} = H_{T_1}'$ or $H_{T_2} = H_{T_2}'$ **then**
22:     $mpc2 \leftarrow mpc2 + 1$
23: **if** $2E \geq k$ **then**
24:     $k \leftarrow 0, E \leftarrow 0, mpc1 \leftarrow 0, mpc2 \leftarrow 0$
25:     $status \leftarrow$ "meeting points"
26: **else if** $k = \widetilde{k}$ **then**
27:     **if** $mpc1 > 0.4k$ **then**
28:         $T \leftarrow T_1$
29:         $k \leftarrow 0, E \leftarrow 0$
30:     **else if** $mpc2 > 0.4k$ **then**
31:         $T \leftarrow T_2$
32:         $k \leftarrow 0, E \leftarrow 0$
33:     $mpc1 \leftarrow 0, mpc2 \leftarrow 0$
34:     $status \leftarrow$ "meeting points"
35: **else**
36:     $status \leftarrow$ "meeting points"
37: **return** $status$
```

Algorithm 14 The process $\psi_{u,v}$

```
$i \leftarrow 1, \psi_{u,v}(1) \leftarrow 0$
for all iterations $i$ from 1 to $100|\Pi|$ do
    if error occurs between $u$ and $v$ during iteration $i$, during any phase then
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + 6C_6$
    else if $\varphi_{u,v}(i) > 0$ then
        $\psi_{u,v}(i+1) = \psi_{u,v}(i) + (5C_6+5)X_{i,u,v} + 5C_6(Y_{i,u,v}) - 5(1 - Y_{i,u,v})$
    else
        $\psi_{u,v}(i+1) = \psi_{u,v}(i)$
```

h_1, note that $\psi_{u,v}$ increases by $(5C_6+5)X_{i,u,v} + 5C_6 Y_{i,u,v} - 5(1-Y_{i,u,v}) = 5C_6 + 5 - 5(1) = 5C_6$, and in the case of a hash collision in h_2, which only occurs in the absence of a hash collision in h_1, we get that $(5C_6+5)X_{i,u,v} + 5C_6 Y_{i,u,v} - 5(1-Y_{i,u,v}) = 5C_6$. In the absence of a hash collision in either, we have that $X_{i,u,v} = Y_{i,u,v} = 0$, and hence $\psi_{u,v}$ decreases by 5. The remainder of the argument is identical to the proof of Lemma 2.2.15.

We will let D denote the number of triples (i,u,v) such that $\psi_{u,v}(i) > 0$ at the beginning of the iteration. Let $D' \leq D$ denote the number of such triples (i,u,v) where, addititionally, there is no error between u and v in i. Let $D_2 \leq D'$ denote the number of triples (i,u,v) where, in addition to this, there is no hash collision in h_1 in (i,u,v). These are the triples where it is possible to have a hash collision in h_2, but there is no error. We start with the following proposition:

Proposition 2.5.2. *Let D' be defined as above. The number of hash collisions in h_2 in triples without errors is at most $2p_2D'$ with probability $\geq 1 - \exp(-\Omega(p_2D'))$.*

Proof. Note that hash collisions in h_2 can only occur in triples in D_2 or in triples (i,u,v) with an error between u and v in iteration i. In each of the D_2 triples (i,u,v), the event of a hash collision in h_2 is either iid $\text{Ber}(p_2)$ or $\text{Ber}(0)$, where the latter can happen if $T_{u,v}(i) \neq T_{v,u}(i)$ but $\psi_{u,v}(i) > 0$.

Instead, consider the sum of a sequence of D' $\text{Ber}(p_2)$ random variables. This stochastically dominates the number of hash collisions in h_2 in triples without errors, since $D_2 \leq D'$ by definition. The statement follows from a standard Chernoff bound applied to the sum. □

Lemma 2.5.3. *Let* Err *denote the number of errors that the non-oblivious adversary commits in an execution of RobustProtocolV2 (Algorithm 12), and let* CC *denote the communication complexity in the execution. Then with probability* $1 - \exp(-\Omega(|\Pi|))$*, we have that either* $\mathsf{CC} \leq 200\alpha|\Pi| m \log\log(m)$*, or that* $\mathsf{Err} > \frac{\varepsilon}{m \log\log(m)}\mathsf{CC}$.

Proof. Recall that we set $K = m \log\log(m)$. We will simply bound the upper probability that $\mathsf{CC} > 200\alpha|\Pi|K$ and simultaneously $\mathsf{Err} \leq \frac{\varepsilon}{K}\mathsf{CC}$. We invoke Lemma 2.5.4 to establish that these two events happen only if $D > \beta \cdot \mathsf{Err}$, where $\beta = \frac{CC}{3\alpha K \mathsf{Err}} \geq \frac{1}{3\alpha\varepsilon}$. Note that this holds for any adversary, oblivious or not.

Lemma 2.5.4 (Reformulation of Lemma 2.2.14 from [58])**.** *Consider a run of Algorithm 12. Denote the number of dangerous triples in this run by D, and the number of errors by* Err. *Suppose that the communication complexity in this run satisfies* $\mathsf{CC} > 200\alpha|\Pi|K$*, where* α *is the constant multiplying the communication in Lemma 2.2.3, and suppose that* $\mathsf{Err} \leq \frac{\varepsilon}{K}\mathsf{CC}$.

If $\mathsf{Err} > 0$*, then* $D \geq \beta \cdot \mathsf{Err}$*, where* $\beta \overset{\text{def}}{=} \frac{\mathsf{CC}}{3\alpha K \mathsf{Err}} \geq \frac{1}{3\alpha\varepsilon}$*. If* $\mathsf{Err} = 0$*, then* $D = 0$ *trivially.*

Now our task is to establish that

$$\mathbb{P}[D > \beta \mathsf{Err}] \leq \exp(-\Omega(|\Pi|)) \tag{2.44}$$

for any adversary, which will bound the communication complexity of the protocol for any adversary. First, we note that if $D > \beta\mathsf{Err}$, then D', the number of triples that additionally have no error, is at least $(\beta - 1)\mathsf{Err}$. Therefore, Proposition 2.5.2 tells us that the number of collisions in the hash h_2 (in triples without errors) is at most $2p_2D'$ with probability at least $1 - \exp(-\Omega(p_2D')) \geq 1 - \exp(-\Omega(p_2|\Pi|))$, for any adversary, where we use the fact that $D' \geq (\beta - 1) \cdot \mathsf{Err} = \Omega(\mathsf{CC}/K) = \Omega(|\Pi|)$.

We can use this to condition on the event that the number of hash collisions in h_2 is at most $2p_2D'$, without losing too much in our bound. Formally, Proposition 2.5.2 gives us that

$$\mathbb{P}[D > \beta\mathsf{Err}] \leq \mathbb{P}[(D > \beta\mathsf{Err}) | (\#\text{ hash col in } h_2 \leq 2p_2D')] + \exp(-\Omega(p_2|\Pi|)) \tag{2.45}$$

for any (non-oblivious) adversary. So it suffices to show that, conditioned on the number of hash collisions in h_2 being at most $2p_2D'$, no adversary can make $D > \beta\mathsf{Err}$ except with negligible probability, where the probability is taken over both the CRS and the privately sampled randomness.

This is done by showing this fact for any adversary that additive and *oblivious to collisions in* h_1, this occurs with probability at most $\exp(-\Omega(\beta \cdot \mathsf{Err} \cdot \log(m)))$(shown below in Lemma 2.5.6).

To establish the result for arbitrary adversaries, we union bound the above probability over all possible oblivious, additive adversaries that induce more than errors and show this is $\exp(-\Omega(\varepsilon|\Pi| \cdot \log(m)))$. This is very similar to what we did in Section 2.3.4. The number of oblivious, additive adversaries that induce exactly Err errors is at most

$$\binom{Cm^2 \log\log(m)|\Pi|}{\mathsf{Err}} \cdot 2^{\mathsf{Err}} \tag{2.46}$$

for some constant C. For a proof of this fact, see the proof of Claim 2.3.34; the proof of this fact is identical, with the $\log(m)$ replaced by $\log\log(m)$ due to the fact that we take $K = m\log\log(m)$ in this section. Now, we separate the remainder of the proof into cases.

Case 1: $\mathsf{Err} \leq 200\alpha\varepsilon|\Pi|$. In this case, we use the montonicity of binomial coefficients where the bottom is much smaller than the top to upper bound the RHS of Eq. (2.46) by $\binom{Cm^2 \log\log(m)|\Pi|}{200\alpha\varepsilon|\Pi|}$. Then, we apply the inequality that $\binom{n}{k} \leq \left(\frac{ne}{k}\right)^k$ to get that the number of such adversaries is at most $\exp(O(\varepsilon|\Pi| \cdot \log(m)))$.

Hence, the probability that there exists an oblivious adversary that makes at most $200\alpha\varepsilon|\Pi|$ errors and manages to make $D > \beta\mathsf{Err}$ (conditioned on the number of hash collisions in h_2 being at most $2p_2D'$) is at most

$$\sum_{\mathsf{Err}=1}^{200\alpha\varepsilon|\Pi|} \exp(O(\varepsilon|\Pi| \cdot \log(m))) \cdot \exp(-\Omega(\beta \cdot \mathsf{Err} \cdot \log(m)))$$

Using the fact that $\beta \cdot \mathsf{Err} = \Omega(|\Pi|)$ by the definition of β, we can simplify the above to

$$\exp(-\Omega(|\Pi|\log(m))),$$

as desired.

Case 2: $\mathsf{Err} > 200\alpha\varepsilon|\Pi|$. In this case, we directly apply the inequality that $\binom{n}{k} \le \left(\frac{ne}{k}\right)^k$ to get that the number of such adversaries is at most $\exp(O(\mathsf{Err} \cdot \log(m)))$.

Hence, the probability that there exists an oblivious adversary that makes at most $200\alpha\varepsilon|\Pi|$ errors and manages to make $D > \beta\mathsf{Err}$ (conditioned on the number of hash collisions in h_2 being at most $2p_2D'$) is at most

$$\sum_{\mathsf{Err}=200\alpha\varepsilon|\Pi|+1}^{Cm^2\log\log(m)|\Pi|} \exp(O(\mathsf{Err} \cdot \log(m))) \cdot \exp(-\Omega(\beta \cdot \mathsf{Err} \cdot \log(m)))$$

Using the fact that $\beta \ge 1/(3\alpha\varepsilon)$, we can simplify this down to

$$\exp(-\Omega(-\beta \cdot \mathsf{Err}\log(m))$$

and using the fact that $\beta \cdot \mathsf{Err} = \Omega(|\Pi|)$, we get

$$\exp(-\Omega(|\Pi|\log(m)))$$

as desired.

Putting the two cases together with Eq. (2.45), we establish Eq. (2.44), completing the proof. □

Lemma 2.5.5. *Let* Err *denote the number of errors that the non-oblivious adversary commits in an execution of RobustProtocolV2 (Algorithm 12), let D denote the number of triples (i,u,v) with $\psi_{u,v}(i) > 0$, and let $\varepsilon^* > 0$ be a fixed constant larger than $4000\alpha C_6\varepsilon$. Then with probability* $1-\exp(-\Omega(|\Pi|))$, *we have that* $\mathsf{Err} \le 200\alpha\varepsilon|\Pi|$ *and* $D \le \varepsilon^*|\Pi|$, *or that* $\mathsf{Err} > \frac{\varepsilon}{m\log\log(m)}\mathsf{CC}$.

We note that the probability in Lemma 2.5.5 hides factors of ε^* in the exponent.

However, since ε^* is not going to 0 with ε, we do not include it in the exponent.

Proof. First we can bound the probability that $\mathsf{Err} > 200\alpha\varepsilon|\Pi|$ and simultaneously $\mathsf{Err} \le (\varepsilon/m \log\log(m))\mathsf{CC}$. These two events occur simultaneously only if $\mathsf{CC} > 200\alpha|\Pi|K$, and we already upper bounded the probability of this event with the desired quantity in Lemma 2.5.3. All that remains is to bound the probability that $\mathsf{Err} \le 200\alpha\varepsilon|\Pi|$ and simultaneously $D > \varepsilon^*|\Pi|$, which will establish the desired result.

Finally, we consider any adversary and upper bound the probability that it can cause $D > \varepsilon^*|\Pi|$, conditioned on the number of errors being at most $200\alpha\varepsilon|\Pi|$. This implies that $D' > (19/20)\varepsilon^*|\Pi|$, and so Proposition 2.5.2 implies that the number of hash collisions from h_2 is at most $2p_2D'$ with probability at least $1 - \exp(-\Omega(p_2\varepsilon^*|\Pi|)) \ge 1 - \exp(-\Omega(|\Pi|))$.

Then, it remains to show that our nonoblivious adversary cannot cause D to be too large conditioned on this event, as conditioning on this event can add at most $\exp(-\Omega(|\Pi|))$ to the overall probability. Similar to the proof of Lemma 2.5.3, we apply Lemma 2.5.6 to get that $\mathbb{P}[D > \varepsilon^*|\Pi|] < \exp(-\Omega(\varepsilon^*|\Pi|\log(m)))$ for any additive adversary that is *oblivious* to collisions in h_1 and commits at most $200\alpha\varepsilon|\Pi|$ errors.

Doing the same union bound over all oblivious, additive adversaries that make at most $200\alpha\varepsilon|\Pi|$ errors as done in the proof of Lemma 2.5.3 proves that no adversary can cause $D > \varepsilon^*|\Pi|$ with probability more than $\exp(-\Omega(|\Pi|))$. Formally, there are at most $\exp(O(\varepsilon|\Pi| \cdot \log(m)))$ adversaries that commit at most $200\alpha\varepsilon|\Pi|$ errors. Hence, the probability that any nonoblivious adversary that commits at most $200\alpha\varepsilon|\Pi|$ errors can cause $D > \varepsilon^*|\Pi|$ (conditioned on the number of hash collisions in h_2 being at most $2p_2D'$) is at most

$$\sum_{\mathsf{Err}=1}^{200\alpha\varepsilon|\Pi|} \exp(O(\varepsilon|\Pi| \cdot \log(m))) \cdot \exp(-\Omega(\varepsilon^*|\Pi|\log(m))) = \exp(-\Omega(|\Pi|\log(m))),$$

which completes the proof. □

Lemma 2.5.6. *Suppose that the number of hash collisions in h_2 in triples without errors is at most $2p_2D'$, where p_2 is the hash collision probability of the second hash function. Let $p_1 = \frac{1}{m^{\Theta(1)}}$*

be the hash collision probability of h_1*. Then, fix an arbitrary* oblivious *adversary, and let* k *be a number larger than* $10C_6$*. Then*

$$\mathbb{P}[D > k \cdot \mathsf{Err}] \le \exp(-\Omega(k \cdot \mathsf{Err} \cdot \log(m)))$$

Proof. Fix the errors that the adversary commits, then consider running RobustProtocolV2 (Algorithm 12). Suppose that this run has $D > k \cdot \mathsf{Err}$. Recall that $D' \le D$ be the set of triples (i, u, v) where $\psi_{u,v}(i) > 0$ and no error occurs between u and v in i. Since the number of errors is Err, we get that $D' \ge (k-1) \cdot \mathsf{Err}$. Now we argue that the fraction of hash collisions required to be in D' must be very large, and that this occurs with very low probability. Denote the set of these D' triples as $\widetilde{\mathsf{D}^*}$ for concreteness.

Due to the nonnegativity of $\sum \psi_{u,v}$, this implies that

$$\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} (5C_6+5)X_{i,u,v} + 5C_6(Y_{i,u,v}) - 5(1-Y_{i,u,v}) \ge -6C_6\mathsf{Err}$$

Simplifying and using the fact that $|\widetilde{\mathsf{D}^*}| = D' > 6C_6\mathsf{Err}$, we get that

$$\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} (X_{i,u,v} + Y_{i,u,v}) \ge \frac{4}{5C_6+5}D'.$$

Finally, using the assumption that $\sum_{S'} Y_{i,u,v} \le 2p_2 D'$, and by taking p_2 to be a sufficiently small constant such that $p_2 < 1/(10C_6+10)$, this implies that

$$\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \ge \frac{3}{5C_6+5}D'.$$

So we can upper bound $\mathbb{P}[D > k\mathsf{Err}]$ by upper bounding the probability that $\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \ge \frac{3}{5C_6+5}D'$. Using a Chernoff bound, we get

$$\mathbb{P}\left[\frac{1}{D'}\sum_{(i,u,v)\in\widetilde{\mathsf{D}^*}} X_{i,u,v} \ge p_1 + \frac{2}{5C_6+5}\right] \le \exp\left(-\Omega\left(k\mathsf{Err}\log\frac{1}{p_1}\right)\right).$$

By using the fact that $\log(1/p_1) = \Theta(\log(m))$, we conclude the result. ◻

Finally, we can put Lemma 2.5.3 and 2.5.6 together to prove Theorem 2.5.1.

Proof of Theorem 2.5.1. Due to Lemmas 2.5.3 and 2.5.6, with probability $1 - \exp(-\Omega(|\Pi|)$, we can conclude that $\mathsf{CC} \leq 200\alpha K|\Pi| = O(\mathsf{CC}(\Pi))$ and that the number of errors and hash collisions in the protocol is bounded by $\mathsf{EHC} \leq O(\varepsilon^*|\Pi|)$, where $\varepsilon^* > 0$ is a constant that is larger than ε but is sufficiently smaller than C_7.

Finally, we recall that the potential ϕ rises by at least K in each iteration, and there are $100|\Pi|$ iterations (Lemma 2.2.3). Claim 2.2.19 yields the final result. ◻

Chapter 3

Efficient Reconstruction of Stochastic Pedigrees

3.1 Introduction

With this chapter, we start the portion of our thesis on inference. The setting for this chapter is inspired by biology: given access to genetic data from multiple parties, the central party would like to infer their mutual family relations, or their *pedigree*. We model this as a statistical inference question and provide an efficient algorithm for reconstructing pedigrees that are generated stochastically. Our model of pedigree inference draws from previous work [154, 159].

3.1.1 Motivation

The decreased costs of sequencing technologies have enabled large-scale, data-driven analyses of genomes [82]. Recent science and news articles feature stories only possible due to this plethora of data, such as the recent identification and capture of a high-profile criminal [97] predicated on DNA evidence. In this effort, an individual's genetic information was compared to a large, curated database called GEDMatch consisting of over one million individual genomes. In comparison, there exist databases which are of several orders of magnitude larger in size such as MyHeritage ($\sim$3.7 million [121]), 23andMe ($\sim$10 million [1]), and Ancestry ($\sim$15 million [8]).

This raises the question: how much kinship information can be learned from DNA? Current databases already contain a considerable amount of this information. Indeed, it is estimated that a given US individual of European ancestry, on average, has a third cousin or closer who is already in the MyHeritage database [45]. However, such databases are still far from complete. This calls into question the ability to detect missing kinships based on individuals already present in the database.

This discussion also highlights the issue of *genomic privacy*. Indeed, it becomes much easier to identify and locate individuals by combining the genetic and genealogical information with outside information (addresses, e-mails, family photos, etc.). This potential, having already been demonstrated by the resolution of the aforementioned criminal case, was brought to attention by [45]. From this point of view, the ability to reconstruct genealogies from collected genetic data is of concern for individuals whose information is revealed, even if one has *never* been sequenced. Since our work establishes a positive result in a pessimistic scenario where we start with no ground truth information, we believe that our work brings to attention this critical issue via a theoretical framework.

3.1.2 Our contributions

Without any prior knowledge about the ground truth, can we learn *everyone's* genealogy using their genetic information? In this paper, we study the inference problem of recovering ancestral kinship relationships of a population of *extant* (present-day) individuals, using only their genetic data. Our goal is to use this extant genetic data to recover the *pedigree* of the extant population, under an idealized model. A pedigree is a graph whose nodes (individuals) have edges that encode parent-sibling relationships. The topology and reconstruction of pedigrees are well-studied in bioinformatics from both a theoretical and empirical perspective, and in general the study of pedigrees poses formidable computational and statistical challenges.

In this paper, we introduce a novel recursive algorithm Rec-Gen for pedigree reconstruction. To demonstrate the effectiveness of our approach, we give a mathematical proof that for an idealized generative model on pedigrees, our algorithm is able to approximately recover the true, unknown pedigree only using the genetic data of the extant population. In terms of *sample complexity*, which for our purposes refers to the common gene sequence length of an extant individual, our algorithm greatly outperforms the naive reconstruction method (estimate pairwise distances between the extant individuals, then construct the pedigree that produces these distances). We propose our approach in this work as a prototype for the future study of more general pedigrees, including those involving real-life genetic data, from both a theoretical and empirical perspective. For further discussion on our model of pedigree generation, as well as its features and limitations, see Section 3.1.4 and Section 3.1.6.

3.1.3 Related works

A common method in theoretical evolutionary biology is to model lineages and inheritance via a family of directed acyclic graphs. One line of work is that of *phylogenetics* (refer to [147] for an overview) which uses trees to model the occurrence of large-scale *speciation events* in evolutionary biology. Another line of work is *coalescent theory*, which focuses on variable-height inheritance trees between genes as its main statistic to infer large-scale *population sizes*, as in e.g. [92]. In contrast, pedigrees capture small-scale *individual genealogies* that encode familial relationships. Specifically, most pedigree models are for human genealogies, where we designate exactly two parents to each individual. By construction, such graphs are no longer trees and warrant different strategies for inference. We note that continuous-time random mating models are often used when inferring large-scale population sizes, for example in [92] as well as [91]. This follows from Wright-Fischer dynamics; we direct the reader to [22] for more details about this modeling assumption for populations.

[154] posed the formal definition of pedigrees using graph-theoretic language. In

that work, the authors gave combinatorial arguments proving that one can reconstruct complete pedigrees, assuming the correct ancestral history is provided as an input for each extant individual. Our definition of pedigrees is essentially the same as the one outlined by these authors, though we make the simplification that we do not identify the vertex set bipartition (corresponding to the biological sex of the individuals).

To tie in more closely with real-world applications, one must consider the challenge of estimating these histories from data. Along these lines, [159] studied stochastic processes that one can associate with the pedigree, in such a way that one can prove negative results (information-theoretic impossibility) or positive results (an algorithm) for the reconstruction of the pedigree from extant data. The stochastic process used to show their positive result was based on a very specific family of Markov chains which allows for inference but is quite different from our model.

For the problem of performing pedigree reconstruction on real data, there is a wealth of literature [160, 95, 75, 161, 76, 149, 81, 164]. Such studies apply heuristics that take into account various complications and phenomena observed in human genomes, such as varying levels of correlations between different sites and the presence of mutations that are not inherited from parents.

One line of work particularly relevant to this paper is [75, 76] in which the authors also tackle the problem of pedigree reconstruction from real extant genetic data. Assuming answers to queries of the form, "how much DNA did i and j simultaneously inherit from their ancestors?", they design a statistical test that distinguishes between siblings, half-siblings and cousins. Their method leverages this information with a maximal-clique finding algorithm to iteratively reconstruct the parents, layer-by-layer. There is no proof of correctness provided, but they provide benchmarks on real and simulated data to provide experimental justification. Our contributions have a slightly different flavor: using a similar iterative strategy but with a different statistical test (the novel part of our algorithm) and for a more optimistic set of assumptions, one can actually *provably* reconstruct the pedigree correctly in a sample-efficient way, in an asymptotic sense.

The authors of [76] specifically emphasize their method's ability to reconstruct half-siblings. Technically speaking, this is not allowed in our model and therefore it may appear to the reader that there is something too restrictive or suboptimal about our analysis. One major difference between our model and the aforementioned work is that we model *haploid* individuals (one copy of DNA), while in reality humans are *diploids* (two copies of DNA). Furthermore, in our proof, we guarantee reconstruction of monogamous *couples* of haploid individuals – in other words, up to permutation of the two individuals within each couple. It can be observed that given a monogamous pedigree with a haploid model, one can construct a natural, non-monogamous pedigree with a diploid model such that the total variation of the extant data of the two pedigrees is zero. Therefore, we think that our results should also hold for a diploid model with minor modifications and have correctness guarantees to match the empirical results of the aforementioned work [76], for example by interpreting Fig. 3-1(a) as a pair of diploid half-siblings.

Our work is also closely related to the problem of phylogenetic reconstruction [44, 115, 117, 42]. In this setting, symbols are passed from the root of a phylogenetic tree to descendants via a Markov process such as in the Cavender–Farris–Neyman model, a basic model for mutations. Similar to our inference problem in this work, in phylogenetic reconstruction, one is tasked with reconstructing the tree given only the symbols at the leaves. The main result of [44] characterizes the *sample complexity*—the minimal string length of the data at the leaves such that reconstruction is possible—as logarithmic in the depth of the tree, a phenomenon that our results suggest also holds for the pedigree reconstruction problem. The work [117] provides theoretical guarantees for the problem of learning the phylogenetic generative model (*i.e.*, the topology of the tree as well as the transition matrices), which includes hidden Markov models as a special case, from the extant data under a spectral assumption on the transition matrices (see also later work of [80]). Most closely related to our approach in this paper is the work [115], which shows how to recursively reconstruct phylogenies using techniques from the theory of broadcast processes on trees (see also [42]). This approach provides inspiration for our main algorithm Rec-Gen, which uses similar techniques to recursively reconstruct pedigrees. We direct the reader to [46] and [113] for studies of broadcast processes on trees with binary and large alphabet respectively, and [112] for a generalization to directed acyclic graphs.

3.1.4 Model description and results

We now give an informal, detailed description of our framework for pedigree reconstruction, with a more detailed treatment of the generative model in Section 3.3. Our generative model on pedigrees consists of two parts: a parametric model for generating the network structure on the set of ancestors and extant individuals, and an inheritance procedure for transmitting genetic data from the *founders*, the oldest individuals in the pedigree, to the extant population.

To generate the pedigree network structure, we begin with a large founding population of size N_T. The founders randomly mate monogamously, and each couple gives birth to a random number of children, so that the average number of offspring per couple is a constant[1] α. This procedure of random monogamous mating continues for T subsequent generations, eventually yielding the extant nodes and a pedigree $\mathcal{P}$ formed by the individuals in generations $0, 1, \dots, T$, with N_i nodes at each level i.

Next we describe how genetic data transmits from the founding population to the extant. Every individual in the pedigree has a gene sequence consisting of B symbols placed in B distinct blocks. Each individual in the founding population is initialized with independent uniformly random draws from a very large alphabet Σ. Now we state how parents pass down genes to their children. In a given block, a child inherits, with equal probability, either its mother's or its father's symbol in the corresponding block. This procedure repeats for all couples in a given generation and then continues over subsequent generations so that genetic data is iteratively transferred through the pedigree, eventually giving rise to the gene sequences of the extant individuals.

[1]More precisely, each couple has a random number of children distributed as a Poisson random variable with expectation α.

Our main result is summarized in the following theorem. See Theorem 3.6.1 for a formal statement.

Theorem 3.1.1 (Main result, informal)**.** *Let α and β denote sufficiently large absolute constants independent of N_T, the size of the founding population. Let ε denote a sufficiently small absolute constant independent of N_T. Assume that the alphabet size $|\Sigma|$ is very large with respect to N_T.*

Then given extant genetic data produced from the generative model with alphabet Σ, growth rate α, gene sequence length $B = \beta \log N_T$, and number of generations $T = \varepsilon \log N_T$ as described above, the algorithm Rec-Gen *recovers* 90% *of the true pedigree in every generation, with high probability. Moreover, this algorithm runs in polynomial time in the size of the pedigree and the number of blocks per extant individual.*

Let $\mathcal{P}$ denote the true, unknown pedigree. Our formal version of Theorem 3.1.1 (see Theorem 3.6.1) implies that with high probability Rec-Gen outputs a reconstructed pedigree $\hat{\mathcal{P}}$ whose size is at least $0.9N_i$ in each generation $i \in \{0, \ldots, T\}$, such that every node $\hat{u} \in \hat{\mathcal{P}}$ can be identified with exactly one node $u \in \mathcal{P}$, and this identification preserves relationships in the sense that $\hat{u}$ is a child of $\hat{v}$ in $\hat{\mathcal{P}}$ if and only if u is a child of v in $\mathcal{P}$. In graph-theoretic terminology, our reconstruction $\hat{\mathcal{P}}$ is a (very large) induced subgraph of the truth $\mathcal{P}$.

We note that the stipulation that we recover 90% of the nodes at each level is actually a simplification; in fact, we can make the fraction of reconstructed nodes in each generation *arbitrarily large* by taking α to be large enough. We refer the reader to Theorem 3.6.1 for details.

3.1.5 The Rec-Gen algorithm

The algorithm Rec-Gen consists of a recursive procedure that uses only the genetic information from the extant population to construct a good approximation for the true pedigree $\mathcal{P}$ of depth T that generated the observations. In the first phase of recursion, the algorithm reconstructs the parents of the extant nodes, which we label as the 1st generation. In the t^{th} phase, the algorithm adds a t^{th} generation to the partially reconstructed version of the true pedigree given by the output of the previous phase. The algorithm terminates after T phases of recursion, producing a pedigree $\hat{\mathcal{P}}$ with T generations that well-approximates the true, unknown pedigree $\mathcal{P}$.

We next give a simplified version of our recursive procedure that serves to illustrate the main ideas. See Section 3.6 for a detailed description of Rec-Gen. Suppose that we have constructed a pedigree $\hat{\mathcal{P}}_t$ of depth t, and recall that B refers to the length of the gene sequence of an individual. Also recall that a *couple* refers to a pair of mated individuals.

Note that the first step of our recursive procedure equips each couple with an empirical gene sequence of length B where each block can contain *two* distinct symbols. This empirical gene sequence is constructed based on extant data and should be thought of as determining which symbols belong to at least one of the individuals from the couple in a given block. Also, we say that three gene sequences $\sigma, \sigma', \sigma''$ *overlap* in a block if all three sequences have some symbol in common in that block.

Perform the following steps to output a pedigree $\hat{\mathcal{P}}_{t+1}$ of depth $t+1$.

(1) Collect-Symbols For each couple c in generation t of $\hat{\mathcal{P}}_t$, use the extant genetic data to recover symbols that belong to c as follows.

   – Recover a symbol σ in block $b \in [B]$ of c if c has three extant descendants descended from distinct children of c that all share symbol σ in block b.

   – Repeat this procedure to recover at most one other symbol $\sigma' \neq \sigma$ for c in block b.

(2) Test-Siblinghood For every triple of couples $c, c', c'' \in \hat{\mathcal{P}}_t$ in generation t, determine c, c', c'' to be (mutually) 'siblings' if and only if at least $0.21B$ of their recovered symbols mutually overlap.

(3) Assign-Parents For every maximal collection $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$ of couples in generation t such that every triple in $\mathcal{C}$ consists of mutual siblings, construct a pair of parents in generation $t+1$ that have as children precisely one individual from each

couple in $\mathcal{C}$.[2]

After T iterations of the above recursive procedure, we output a pedigree $\hat{\mathcal{P}}_T$ that gives a good approximation to the underlying pedigree that generated the extant genetic data as described in Theorem 3.1.1. We remark that working with triples as above greatly simplifies our analysis, as discussed in Section 3.2.3[3].

3.1.6 Model discussion and future directions

Our generative model imposes various constraints on the typical pedigrees that we consider. We discuss these modeling assumptions here and also consider the problem of investigating more general models that could more accurately capture properties of real-world data.

First, we consider the assumption that the size of the alphabet Σ is very large with respect to the size N_T of the founding population. Since a "block" represents the unit of inheritance from a parent[4], this implies that with very high probability all of the founders have distinct symbols in their gene sequences, and no two founders share a common symbol.[5] Our large alphabet assumption is equivalent to the assertion that the founders are unrelated.

Second, the stochastic process describing inheritance in our model has the following biological interpretation. A standard concept in population genetics refers to long-running sequence matches as being *identical by descent* (IBD) if they arose due to inheritance from a common ancestor [161]. In contrast, the term *identity by state* refers to the event that two identical tracts in the genome arose by coincidence – via mutations – in two unrelated individuals. Our inheritance model contains the assertion that each block corresponds to true IBD sequences: if two individuals have the same symbol, we can always identify a common ancestor that gave rise to these symbols.

[2]We perform this step in such a way that every child is assigned at most 2 parents.

[3]We note that triples in this work refer to tuples of three nodes in the pedigree; this has nothing to do with the triples in Chapter 2

[4]Using biology terminology, each block can be considered as an idealized abstraction of a collection of *single-nucleotide polymorphisms* (sites of variation) with high *linkage disequilibrium* (empirical measure of correlation) that are passed from parent to child.

[5]Mathematically, this can be thought of as an improper prior on a countably infinite alphabet Σ.

Third, we recall the hypothesis that every couple has on average α children, where α is a sufficiently large absolute constant independent of the size N_T of the founding population. This ensures that, roughly speaking, every new generation is a factor $\alpha/2$ larger than the previous one. Assuming roughly uniform growth of generations, it is necessary that $\alpha > 0$ — otherwise the population would die out and there would be no extant nodes after T generations. More subtly, it is necessary that $\alpha \geq 2$ — otherwise, via standard results from the theory of branching processes (see, *e.g.* [94]) a founding node has a very low probability of passing on its symbols to the extant. In this situation, even *detection* of such an ancestor from extant genetic data alone is information-theoretically impossible. On the other hand, our assumption that α is a large constant essentially amplifies the signal sent from a founder to the extant, and this simplifies our mathematical analysis.

Our first open question considers relaxing the previously discussed assumptions.

Question 4. *What theoretical guarantees can be established for pedigree reconstruction in the context of our generative model when α is very close to 2? What about when the size of the alphabet Σ is finite? Can we analyze more generic models of inheritance where blocks are not inherited i.i.d. from parents?*

A more subtle consequence of our generative model is *inbreeding*, a term we use to refer to the following phenomena: (1) the presence of multiple lowest common ancestors for a pair of extant nodes, and (2) the presence of mated couples such that the two individuals in the couple have a lowest common ancestor (LCA) (see Definition 3.3.5 for the formal definition of an LCA). The *degree* of inbreeding qualitatively refers to the frequency of such structures in the pedigree. Moreover, inbreeding as in (2) is mathematically equivalent to having cycles in the pedigree. In general, a higher degree of inbreeding makes the pedigree reconstruction problem more difficult and in some cases information-theoretically impossible (see Section 3.2.1 for detailed examples). Our choice of model allows for some degree of inbreeding, and our algorithm and analysis are carefully tailored to circumvent this obstacle.

Other assumptions inherent in our model include that the pedigree is *graded, i.e.*, couples are formed from individuals in the same generation, and *monogamous*: a given individual only mates with one other individual. Furthermore, *mutations* — errors in the transmission of genetic data from parents to offspring — are a central component in biological applications that our current model does not incorporate.

Question 5. *What theoretical guarantees can be established for reconstruction of pedigrees in generative models with some combination of (i) a higher degree of inbreeding, (ii) mutations, (iii) non-monogamous mating, and (iv) inter-generational mating?*

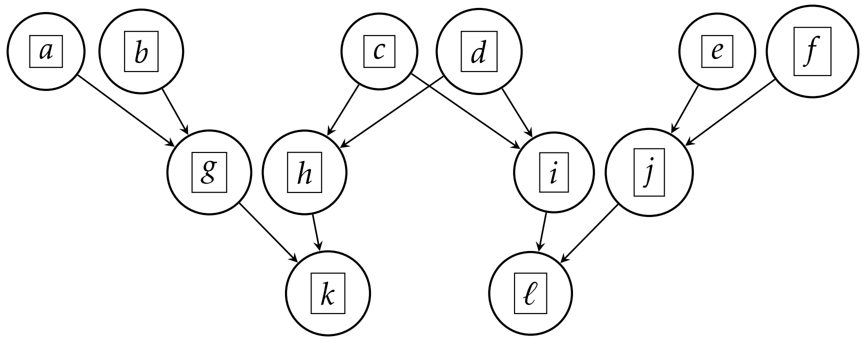
3.2 Inference challenges and techniques

In this section, we detail some of the challenges posed by the reconstruction of pedigrees constructed from our generative model as well as our techniques and analysis for handling them. To develop some intuition for our strategy, we first illustrate some of the properties of pedigrees using concrete examples.

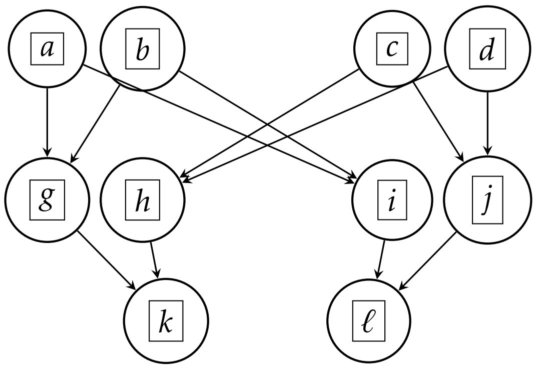
3.2.1 Examples: complications from inbreeding

Recall that two individuals u, v that share the same set of parents are *siblings*. If two individuals share a common subset of grandparents (but not parents), we refer to them as *cousins*.

First consider the pedigrees displayed in Fig. 3-1(a). An important statistic for determining relationships is the correlation between symbols of nodes at the same level. Consider the event E that the left extant shares the same symbol as the right extant. Note that these two extant nodes are cousins sharing a single set of grandparents. The grandparents are the founders in this example, so we assign to each of them a unique symbol ($a \neq b \neq c \neq d \neq e \neq f$). The occurrence of E implies that $k = c$ or $k = d$ via the left extant receiving a symbol from its right parent; this occurs with probability $\frac{1}{2}$. Conditioned on

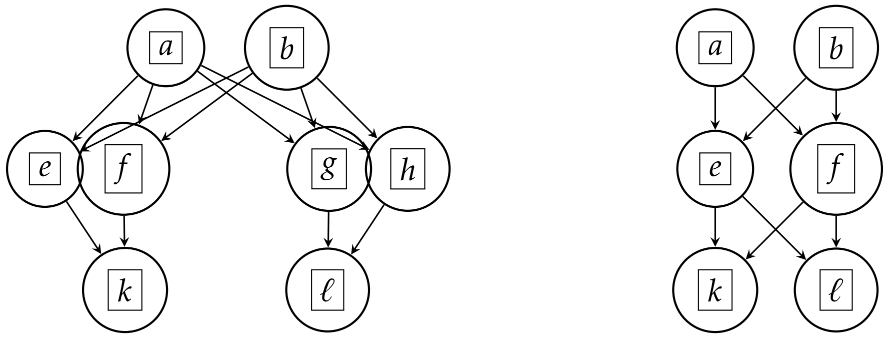


(a) three sets of grandparents (cousins, one way)



(b) two sets of grandparents (cousins, two ways)

Figure 3-1: *Simple examples of depth-3 complete pedigrees with a single block. The letters inside the boxes represents the block data. 3-1(a): The overlap probability is* $\mathbb{P}(k = \ell) = \frac{1}{8}$. *3-1(b): An altered version of 3-1(a) with only two sets of grandparents, which yields* $\mathbb{P}(k = \ell) = \frac{1}{4}$.



(a) four siblings begetting cousins.

(b) two siblings begetting siblings.

Figure 3-2: *Two examples of complete pedigrees with inbreeding. The extants in 3-2(a) are cousins, yet they have a coincidence of $\frac{1}{2}$ as if they were generic siblings from unrelated parents. In comparison, 3-2(b) yields $\frac{3}{4}$ which exceeds the coincidence of siblings.*

this occurring, the right extant block ℓ is the same as k with probability $\frac{1}{4}$, so the overall probability that both receive the same symbol is $\frac{1}{8}$.

Compare this to the example shown in Fig. 3-1(b), where the two extant are cousins in two ways (*siblings marrying siblings*). Note that whichever symbol (out of a, b, c, d) that k is, the right grandchild receives the same independently with probability $\frac{1}{4}$. This is an example of a type of inbreeding where two extant nodes have more than one LCA.

The examples in Fig. 3-2 demonstrate how the correlation between extant nodes is boosted due to the presence of inbreeding. Note that in the *generic* case where extant siblings have an ancestral pedigree that is a tree, these individuals have a $\frac{1}{2}$ fraction overlap in their blocks. For comparison, let us compute the probability of coincidence for the two extant nodes in Fig. 3-2(a). The probability that $k = a$, for example, is

$$\mathbb{P}(k = a) = \mathbb{P}(e = f = a) + \frac{1}{2}\mathbb{P}(\{e, f\} = \{a, b\}) = \frac{1}{4} + \left(\frac{1}{2}\right)^2 = \frac{1}{2}.$$

Since k and ℓ inherit symbols independently from their grandparents, the overall probability is

$$\mathbb{P}(k = \ell) = \mathbb{P}(k = \ell = a) + \mathbb{P}(k = \ell = b) = \left(\frac{1}{2}\right)^2 + \left(\frac{1}{2}\right)^2 = \frac{1}{2},$$

which is precisely the probability that two generic siblings inherit the same symbol.

The situation in Fig. 3-2(b) is even more pronounced. The two parents share the same symbol (either a or b) with probability $\frac{1}{2}$ and have different symbols with probability $\frac{1}{2}$. This means that the coincidence probability is now $\frac{1}{2} + \frac{1}{2} \times \frac{1}{2} = \frac{3}{4}$: their correlation between overlaps is much stronger than that of siblings in the generic case.

From the example in Fig. 3-2(a), we conclude that the statistical model of extant data parametrized by pedigrees is unidentifiable. Stated another way, it is information-theoretically impossible to distinguish between siblings and inbred cousins using only extant data. Thus, in order for any algorithm to succeed in reconstructing a large fraction of the pedigree using only extant data, it is necessary to bound the amount of inbreeding in the ensemble of pedigrees of interest. We accomplish this using a careful analysis of our generative model.

3.2.2 Informal analysis of Rec-Gen

In this section, we present a high-level analysis of the Rec-Gen algorithm. Theorem 3.1.1 states that Rec-Gen yields an accurate reconstruction on 90% of nodes for typical pedigrees from our generative model[6]. Note that a formal statement of this theorem, our main result, is given by Theorem 3.6.1, and a complete proof is contained in the upcoming sections.

Suppose we construct a pedigree $\hat{\mathcal{P}}_t$ on t generations that, for simplicity of the discussion, *exactly* matches the true, unknown pedigree $\mathcal{P}$ up to generation t. We show that Collect-Symbols, Test-Siblings, and Assign-Parents applied to $\hat{\mathcal{P}}_t$ provide an accurate reconstruction of 90% of the nodes at generation $t+1$. In the remainder of this section we give a high-level argument that the output $\hat{\mathcal{P}}_{t+1}$ satisfies the following conditions:

(i) every individual $\hat{u}$ in $\hat{\mathcal{P}}_{t+1}$ can be identified with a unique individual u in $\mathcal{P}$ at generation $t+1$,

(ii) at most 10% of the nodes in generation $t+1$ of $\mathcal{P}$ are not identified with an individual

[6]We note again that the 90% is for simplicity of exposition, and in reality we can recover an arbitrarily large fraction of nodes. This is made precise in Theorem 3.6.1.

in $\hat{\mathcal{P}}_{t+1}$, and

(iii) if v is a child of $\hat{u}$ in $\hat{\mathcal{P}}_{t+1}$, then v is a child of u in $\mathcal{P}$.

Recall that for the purposes of reconstruction, we only have access to the genetic data of the extant.

In this discussion, we refer to three couples $c, c', c'' \in \mathcal{P}$ as (mutual) siblings if there exist individuals $u \in c, u' \in c'$, and $u'' \in c''$ such that u, u', and u'' are mutually siblings. A *clique* refers to a collection of couples $\mathcal{C} = \{c_1, \ldots, c_k\}$ such that every triple from $\mathcal{C}$ consists of mutual siblings.

The next two facts are essential to the argument.

(A) If Collect-Symbols recovers symbol σ in block b for a couple c in generation t, then c also has the symbol σ in block b in $\mathcal{P}$ (Claim 3.6.8).

(B) Collect-Symbols recovers at least 99% of the symbols for at least 99% of the couples in generation t (Lemma 3.5.17).

Together, (A) and (B) imply that for 99% of the couples in generation t, our algorithm gets all of the siblings relationships between these couples correct. To see why, we can use a similar calculation as in the first example of Section 3.2.1 to conclude that the average overlap between the symbols of three individuals that are mutually siblings is 25%. By concentration of binomial random variables about their means, it follows that with high probability, all triples of individuals that are mutually siblings in $\mathcal{P}$ have at least 24.9% mutual overlap between their symbols. A simple union bound combined with (A) and (B) implies that for most triples of individuals in generation t that are mutually siblings, the recovered symbols from Collect-Symbols in those individuals' corresponding couples have overlap at least 21%. Hence, Test-Siblinghood infers correct siblinghood relationships for a majority of triples.

Moreover, our siblings test on the recovered symbols does not have any false-positives:

(C) Test-Siblinghood never misclassifies non-siblings as siblings (Lemma 3.6.5).

The next and last key fact argues that our naive assignment of parents to individuals in cliques as in Assign Parents is in fact the correct assignment in a typical pedigree. This property holds with very high probability over our generative model.

(D) Let $\mathcal{C} \subset \mathcal{P}$ denote a clique at generation t in the true pedigree. Then there exists a couple $\tilde{c}$, which we refer to as the *parents of* $\mathcal{C}$, in generation $t+1$ of $\mathcal{P}$ that has exactly one child in every couple of $\mathcal{C}$, and no other couple has more than 1 child in $\mathcal{C}$ (Lemma 3.4.13).

Together, (A), (B), (C), and (D) imply that our reconstruction criteria (i), (ii), and (iii) from the beginning of this section hold, as we now justify. Recall that we already showed (A) and (B) imply that we classify a large fraction of the couples at generation t correctly as siblings. Moreover, part (C) and the transitivity of siblinghood in $\mathcal{P}$ imply that cliques in our reconstruction really correspond to cliques in the truth. By part (D) such cliques have unique parents. Thus, for (i), we identify newly constructed couples $\hat{u} \in \mathcal{P}_{t+1}$ with the *unique* parents $u \in \mathcal{P}$ of the clique formed by the children of $\hat{u}$, further pairing the two individuals in u with those in $\hat{u}$ arbitrarily. With this identification, (iii) follows immediately. To show part (ii), later in the paper we give a sufficient condition for a couple at generation t to have 99% of its symbols collected by Collect-Symbols as in (B) (see Lemma 3.5.17). Then we show that 90% of individuals in generation $t+1$ have children in such couples (see Proposition 3.5.16), which proves part (ii). Essentially, this sufficient condition amounts to saying that a couple c at generation t has no inbreeding (cycles) above or below it (*i.e.* among its ancestors or descendants, respectively) and that the pedigree of descendants of c contains a $\alpha/4$-ary tree (see Definition 3.5.14).

3.2.3 Motivation for using triples

It is tempting to employ a seemingly simpler recursive scheme than the one described in Section 3.1.5 that operates on pairs instead of triples. As an example, consider an alternative recursive procedure such that:

1. Collect-Symbols only uses **pairs** of extant descendants to recover symbols of a couple c,
2. Test-Siblinghood considers only **pairs** of couples at generation t and detects them to be siblings if their strings overlap by at least 49%, and
3. Assign-Parents assigns parents to individuals in maximal collections $\mathcal{C}$ such that every **pair** of couples is (tested as) siblings.

Unfortunately, this simpler approach encounters two major technical complications.

First, working with a pairwise siblings test introduces a problem for the step of assigning parents. Define a *pairwise clique* to be a collection of couples so that every pair of couples passes the pairwise siblings test. With high probability, it turns out in every generation there exist a constant number of pairwise cliques that are not explained in the naive way of assigning to this clique parents that have precisely one child per couple. In particular, in the true pedigree $\mathcal{P}$ it is possible to have three couples that mutually pass the pairwise siblings test, yet there are **three** distinct parent couples each having precisely two children among these three couples. See Fig. 3-3 for an illustration. This type of structure, though rare, occurs a constant number of times in each generation, and thus introduces inherent errors in our reconstruction that accumulate at every step of iteration.

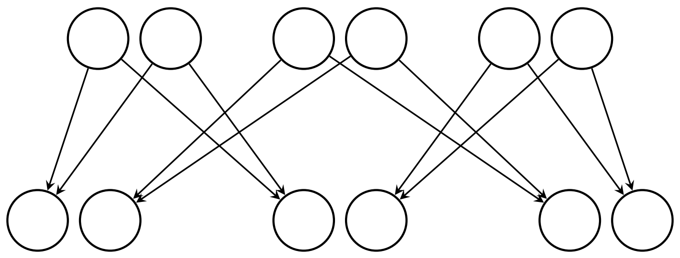


Figure 3-3: *An undesirable subpedigree, where three child couples have mutual siblingship, but they do not mutually share a parent couple.*

A second problem caused by working with pairs arises in the step of collecting symbols. The pairwise version of our algorithm assigns a symbol to a couple if that symbol occurs in two extant descendants that are descended from distinct children of that couple. In our generative model, it turns out that with high probability there are a *logarithmic* number of pairs of extant nodes that have at least two LCA's. For such pairs, the pairwise algorithm does not accurately assign symbols to their reconstructed ancestors. Similar to the previous issue, these errors snowball and make the analysis for proving Theorem 3.1.1 very difficult.

On the other hand, working with an algorithm using triples as described in Section 3.1.5 makes for a much cleaner analysis and nicer reconstruction guarantee. This innovation circumvents the technical complications of the pairwise version because every clique (recall that this is a collection of couples where every triple consists of mutual siblings) can be explained in a naive way (Lemma 3.4.13), and in our generative model every triple of extant individuals descended from distinct children of a given ancestor have that ancestor as their *unique* LCA with very high probability (Lemma 3.4.16).

3.2.4 Outline of technical arguments

The remainder of the paper, which provides a formal proof of Theorem 3.1.1, is divided into four parts.

- Section 3.3 provides preliminary definitions and a formal definition of our generative model.
- Section 3.4 proves important properties about the typical network structure of pedigrees from our generative model.
- Section 3.5 proves important properties about the block statistics of the extant nodes in a typical pedigree from our generative model.
- Section 3.6 gives a precise description of Rec-Gen and provides a formal statement and proof of Theorem 3.1.1.

Specifically, in Section 3.4 we rigorously quantify the degree of inbreeding in typical pedigrees from our model by counting the number of *collisions* (see Definition 3.4.6 and Lemma 3.4.8). This has several useful consequences, including that every clique has a unique parent (fact (D) from Section 3.2.2, also see Lemma 3.4.13) and that the extant individuals used in Collect-Symbols have a unique LCA (see Lemma 3.4.16). In particular, the latter is key to showing fact (A) from Section 3.2.2.

In Section 3.5, we provide a definition (see Definition 3.5.14) that essentially characterizes the individuals in $\mathcal{P}$ that are reconstructible via Rec-Gen. We show that couples involving such individuals, referred to as *awesome couples*, transmit many of their symbols to the extant, with high probability (see Lemma 3.5.17). In particular, awesome couples have at least 99% of their symbols recovered by Collect-Symbols (fact (B) from Section 3.2.2). We also prove an important result for our siblings test: triples of individuals that are not mutually siblings have mutually overlap at most 19% (see Lemma 3.5.4). This combined with fact (A) from Section 3.2.2 essentially shows that Test-Siblinghood never classifies non-siblings as siblings (fact (C) from Section 3.2.2, see also Lemma 3.6.5).

Our final section, Section 3.6 ties everything together, following fairly closely the high-level argument presented in Section 3.2.2 to prove the formal version of Theorem 3.1.1.

3.3 Preliminaries

3.3.1 Key definitions and terms

Definition 3.3.1. *A* ***pedigree*** $\mathcal{P} = (V, E)$ *is a directed acyclic graph (DAG) with vertices* V *and edges* E *where every vertex has indegree at most* 2*. The collection of vertices of indegree zero are referred to as the* ***founders****, and the collection of vertices of outdegree zero are referred to as the* ***extant****.*

Definition 3.3.2. *If the indegree of each vertex in the underlying DAG is either* 2 *or* 0*, then* $\mathcal{P}$ *is called a* ***complete*** *pedigree.*

In this work, we focus on a special family of complete pedigrees that are both *graded* and *monogamous*.

Definition 3.3.3. *$\mathcal{P}$ is said to be **graded** if the vertices $V(\mathcal{P})$ can be partitioned into $\bigcup_{i=0}^{T} V_i(\mathcal{P})$ such that $V_T(\mathcal{P})$ are the founders, $V_0(\mathcal{P})$ are the extant, and all directed paths $e_T,\ldots,e_1$ from $V_T(\mathcal{P})$ to $V_0(\mathcal{P})$ can be written as a sequence of edges $e_t = (v_t \to v_{t-1})$ where $v_t \in V_t(\mathcal{P})$ and $v_{t-1} \in V_{t-1}(\mathcal{P})$ for each t. The founders' index T is the **depth** of the pedigree.*

*$\mathcal{P}$ is said to be **monogamous** if for every vertex u of outdegree > 0, there exists a unique vertex u' such that $(u \to v) \in E \iff (u' \to v) \in E$. The unordered pair $\{u, u'\}$ is referred to as a **couple**.*

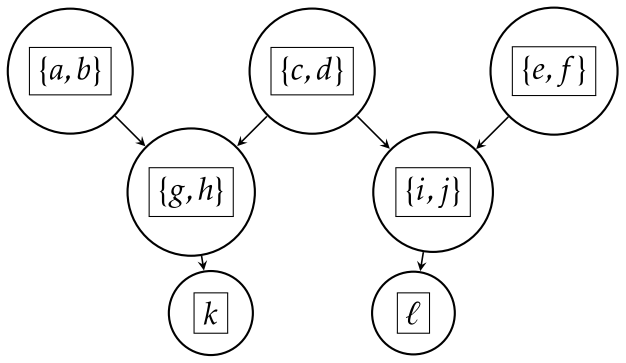
We assume that every non-extant individual in the pedigree is in a couple, and so the number of vertices at each non-extant level is even. This assumption is effectively without loss of generality—if an individual is not in a couple, then it has no descendants, and so we cannot recover information about this individual or even its existence.

An example of a complete, graded, monogamous pedigree is shown in Fig. 3-1(a). In our model, symbols are passed down from parents to children in a completely symmetric way. Thus, given the data of the children, it is impossible to distinguish the owner of each symbol from amongst the two parents. The goal of this paper is to show how one can provably infer the structure of a complete pedigree from extant genetic data via the reconstruction of the ancestral symbols, modulo *block phasing* (determining which symbol belongs to which parent for each block). Therefore, we introduce the following version of a pedigree which condenses this information.

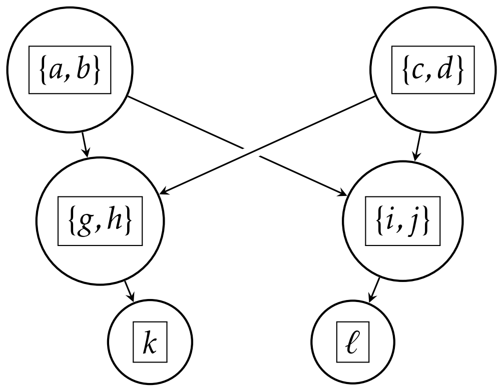
Definition 3.3.4. *A **coupled** pedigree $\mathcal{Q} = (V_\mathcal{Q}, E_\mathcal{Q})$ induced by a complete, monogamous pedigree $\mathcal{P} = (V_\mathcal{P}, E_\mathcal{P})$ is defined as follows:*

- *$V_\mathcal{Q} \subset \binom{V_\mathcal{P}}{2}$ is obtained by merging couples $c = \{u, u'\} \subset V_\mathcal{P}$ into a single node (extant individuals remain singletons), introducing edge multiplicity.*
- *$E_\mathcal{Q}$ is the result of halving the number of resulting copies of each edge after merging couples.*

In particular, a coupled pedigree is also a pedigree. Examples are drawn in Fig. 3-4 in relation to Fig. 3-1, where the complete pedigree 3-1(a) induces a coupled pedigree 3-4(a)



(a) coupled version of 3-1(a)



(b) coupled version of 3-1(b)

Figure 3-4: *3-1(a) induces coupled pedigree 3-4(a), while 3-1(b) induces 3-4(b).*

and 3-1(b) induces 3-4(b).

The only information that is lost after transforming a complete, monogamous pedigree into a coupled pedigree is the block phasing. Indeed, observe that given the coupled structure $\mathcal{Q} = (V_{\mathcal{Q}}, E_{\mathcal{Q}})$, one can easily obtain the individual structure $\mathcal{P} = (V_{\mathcal{P}}, E_{\mathcal{P}})$ up to block phasing as follows: (1) add the extant individuals in $V_0 \subset V_{\mathcal{Q}}$ to $V_{\mathcal{P}}$, (2) for every non-extant node $c \in V_{\mathcal{Q}}$ add individuals u_c, u'_c to $V_{\mathcal{P}}$, and (3) given parents c_1 and c_2 of c in $\mathcal{Q}$, add the four edges $u_{c_1} \to u_c, u'_{c_1} \to u_c, u_{c_2} \to u'_c, u'_{c_2} \to u'_c$ to $E_{\mathcal{P}}$. In addition, if $\mathcal{P}$ is graded, $\mathcal{Q}$ retains a graded structure $V_{\mathcal{Q}} = V_0(\mathcal{Q}) \cup \cdots \cup V_T(\mathcal{Q})$ so that $V_0(\mathcal{Q})$ are the extant nodes and $V_1(\mathcal{Q}), \ldots, V_T(\mathcal{Q})$ are depth-graded couple nodes. In particular, the graph structure of an individuals pedigree $\mathcal{P}$ uniquely determines the graph structure of its associated coupled pedigree $\mathcal{Q}$ and vice versa.

Given the previous discussion, since our goal is to recover the graph structure of an underlying true pedigree $\mathcal{P}$ given gene sequences of a large number of extant individuals, it suffices to reconstruct the associated coupled pedigree $\mathcal{Q}$.

Furthermore, since the graph underlying a pedigree is a DAG, given a subset S of the pedigree, it is natural to consider the notion of "ancestors" (nodes $\mathsf{anc}(S)$ **from** which there is a directed path to S) and "descendants" (nodes $\mathsf{desc}(S)$ **to** which there is a directed path from S). Also for simplicity, we stipulate that every node v is both a descendant and an ancestor of itself, *i.e.*, $v \in \mathsf{anc}(v)$ and $v \in \mathsf{desc}(v)$. Since the indegree of each node can be more than one, it is possible for two nodes to have more than one “lowest common ancestor”. We define this now.

Definition 3.3.5 (Lowest Common Ancestors)**.** *Let S denote a set of nodes in a pedigree $\mathcal{P}$. The set of* ***lowest common ancestors*** *of S, denoted $\mathsf{LCA}(S)$, consists of all nodes $u \in \mathcal{P}$ such that u is an ancestor of every node in S, and moreover, no descendant of u is an ancestor of every node in S.*

During our analysis, we often restrict our attention to the information that the pedigree contains about the ancestors or descendants of a particular collection of nodes. In particular, we want to exploit (sub)structures that are not too intertwined. The following definitions make these ideas precise:

Definition 3.3.6 (Subpedigrees)**.** *Let $W \subset V_{\mathcal{P}}$ denote a subset of nodes of pedigree $\mathcal{P}$. The subgraph of $(V_{\mathcal{P}}, E_{\mathcal{P}})$ induced by W is itself a pedigree, which we call the* ***subpedigree*** *of $\mathcal{P}$* ***induced by*** *W.*

Definition 3.3.7 (Ancestral pedigrees)**.** *Let $W_k \subset V_k(\mathcal{P})$ denote a subset of vertices at level k of a graded pedigree $\mathcal{P}$. The subpedigree induced by $W_k \cup \mathsf{anc}(W_k)$ is the (level k)* ***ancestral subpedigree*** *of $\mathcal{P}$ induced by W_k.*

Definition 3.3.8 (Descendant pedigrees)**.** *Let $W_k \subset V_k(\mathcal{P})$ denote a subset of vertices at level k of a graded pedigree $\mathcal{P}$. The subpedigree induced by $W_k \cup \mathsf{desc}(W_k)$ is the (level k)* ***descendant subpedigree*** *of $\mathcal{P}$ induced by W_k.*

Definition 3.3.9 (Tree pedigrees)**.** *A pedigree $\mathcal{P}$ that has no undirected cycles (when the directions of the edges in $E_{\mathcal{P}}$ are ignored) is called a* ***tree*** *pedigree.*

Note that coupled pedigrees can have edges of multiplicity two, though only in the case where two siblings form a coupled node, which a rare structure in our generative model. In coupled pedigrees, we consider a double edge to be an undirected cycle of length two. Hence, a tree pedigree consists entirely of simple or multiplicity 1 edges.

As we demonstrate (*e.g.* Lemma 3.5.4), coupled tree pedigrees exhibit a type of correlation decay between blocks that enable us to perform inference on the structure. In contrast, non-tree coupled pedigrees correspond to pedigrees with inbreeding, which can arise in nature and appear in our probabilistic model as well. Section 3.2.1 illustrates examples of such structures. These types of structures introduce challenges for performing inference under our generative model.

3.3.2 Siblings in a pedigree

Note that siblinghood is a transitive relationship: if u, v are siblings and v, w are siblings, then so are u, w. As alluded to in Section 3.2.3, it is important to look at these relationships in *triplets*. We now detail how one can encode this information as a *3-uniform hypergraph*.

Definition 3.3.10. *A* ***3-uniform hypergraph*** *is a pair (V, E) of vertices and a multiset of edges, so that each edge is an unordered triple $\{u, v, w\}$ of vertices in V.*

Definition 3.3.11. *Let $\mathcal{P}$ be a* <u>*coupled pedigree*</u> *of depth T (each non-extant node is a set of a pair of individuals). The* ***siblinghood hypergraph*** *G_k of $\mathcal{P}$ at level $k > 0$ is the 3-uniform hypergraph that describes the three-way sibling relationships of its level-k members. For every triple $e = \{c_1, c_2, c_3\}$, the edge multiplicity $n(e; G_k)$ is*

$$n(e;G_k) = \begin{cases} 0 & \textit{if } \nexists\, (u_1,u_2,u_3) \in c_1 \times c_2 \times c_3 \textit{ such that } u_1,u_2,u_3 \textit{ are siblings} \\ 1 & \textit{if } \exists \textit{ unique } (u_1,u_2,u_3) \in c_1 \times c_2 \times c_3 \textit{ such that } u_1,u_2,u_3 \textit{ are siblings} \\ 2 & \textit{else} \end{cases}$$

The siblinghood hypergraph G_0 is defined similarly, by considering each extant individual u as a degenerate (cardinality 1) couple $c_u = \{u\}$ and applying the above definition (Each hyperedge appears zero or once, never twice).

Recall that a **clique** in a 3-uniform hypergraph is a collection of vertices such that all possible triplets form an edge. The next statement is an observation that follows from the definition of G_k and the transitivity of siblinghood.

Proposition 3.3.12. *If $c_1, \dots, c_m$ are level-k couples that respectively contain individuals $u_1, \dots, u_m$ which are siblings, then $c_1, \dots, c_m$ form a clique in G_k.*

3.3.3 Probability Tools

We denote a Poisson distribution with mean λ as $\mathsf{Pois}(\lambda)$. We use some basic tools from probability theory in our proof. The first is referred to in literature as *Poisson thinning*, see *e.g.* [101].

Proposition 3.3.13 (Poisson Thinning)**.** *Let $N \sim \mathsf{Pois}(\lambda)$, and let $X_1, X_2, \dots$ be iid Ber(p) random variables that are independent of N. Then $X = \sum_{i=1}^{N} X_i$ is $\mathsf{Pois}(\lambda p)$-distributed.*

Second, we recall that sums of Poisson random variables are themselves Poissons:

Proposition 3.3.14. *Fix $N > 0$ and let $X_1, X_2, \dots, X_N$ be iid $\mathsf{Pois}(\lambda)$ random variables. Then $X = \sum_{i=1}^{N} X_i$ is $\mathsf{Pois}(\lambda N)$-distributed.*

Lastly, we will use the fact that Poisson distributions also have sub-exponential tails.

Proposition 3.3.15 (Poisson tail probability)**.** *Let $X \sim \mathsf{Pois}(\lambda)$. Then for any $x > 0$, we have*

$$\mathbb{P}(|X - \lambda| \geq x) \leq 2 \exp\left(-\frac{x^2}{2(\lambda + x)}\right)$$

For a proof, refer to Chapter 2 of [134].

3.4 Structure of Poisson Pedigrees

3.4.1 Model Description

We now describe our simple model for generating a population and its genetic data. The model is best viewed in two stages. In the first stage, we generate the population as well as the pedigree topology $\mathcal{P}_{\mathsf{indiv}}$ on these individuals, and in the second stage, we generate the genetic data given this pedigree structure. Note that the random individual pedigree $\mathcal{P}_{\mathsf{indiv}}$ constructed below is **graded, monogamous,** and **complete**.

Part I: Pedigree topology

1. To generate $\mathcal{P}_{\mathsf{indiv}}$, start with $N_T = N$ founding individuals in V_T and make an arbitrary maximum matching of these individuals to create a set of mated couples. For each couple, generate an independent Pois(α) number of children, where $\alpha > 0$ is a fixed parameter throughout the entire pedigree. These newly generated individuals form the nodes in V_{T-1}.

2. Repeat the above process to generate the individuals in $V_{T-2}, \ldots, V_0$.

Once we have the population and pedigree structure as above, we generate the genetic data in the following manner.

Part II: Inheritance procedure

1. Each individual u in $\mathcal{P}_{\mathsf{indiv}}$ has a length-B string σ_u (u's **gene sequence**). The string's indices are referred to as **blocks**.

2. For each founding individual u in V_T and for each block $b \in [B]$, each $\sigma_u(b)$ is drawn i.i.d. uniformly from an alphabet Σ. For our model, Σ is an infinite-sized alphabet: we simply require that each block of each founder has a unique symbol.

3. Every other individual v in the population has exactly two parents f and m. Conditioned on σ_f and σ_m, independently over $[B]$, the ith block of v copies $\sigma_f(i)$ with probability 0.5 and $\sigma_m(i)$ with probability 0.5.

Remark 3.4.1. *We adopt the following conventions in the remainder of the paper.*

1. *We let $\mathcal{P}$ denote the* ***coupled pedigree*** *induced (see Definition 3.3.4) by the randomly generated individual pedigree $\mathcal{P}_{\mathsf{indiv}}$ constructed in Part I above.*

2. *We use the term* ***coupled node****, or simply* ***node*** *when the context is clear, to refer to a vertex of $\mathcal{P}$. We use the term* ***individual*** *to refer to an element of $\mathcal{P}_{\mathsf{indiv}}$ contained in a coupled node of $\mathcal{P}$. Unless otherwise explicitly noted, parent-child relationships are taken according to the structure of the coupled pedigree $\mathcal{P}$. That is, given $u, v \in \mathcal{P}$ we use the phrase, "u* ***is a child of*** *v," to mean that the couple u contains an individual who is an offspring of the mated couple v. Finally, we say that coupled nodes $u, v \in \mathcal{P}$ are* ***siblings*** *if u and v contain individuals who are siblings in $\mathcal{P}_{\mathsf{indiv}}$.*

3. $\mathbb{P}$ *denotes the probability measure over the randomly generated pedigree $\mathcal{P}$ as well as the random inheritance procedure.*

To given an example of our terminology, there are two individuals in a non-extant coupled node. Each individual is a vertex of $\mathcal{P}_{\mathsf{indiv}}$, and together they form a coupled node, which is a vertex of $\mathcal{P}$. Note that as an artifact of our definitions, extant individuals are both coupled nodes *and* individuals in $\mathcal{P}$. Moreover extant nodes have exactly one parent in $\mathcal{P}$ given by the coupled node containing the individuals comprising that extant individuals biological parents, as determined by our generative model.

To further emphasize the previous remark, recall that by the discussion in Section 3.3.1, there is a unique correspondence between coupled pedigrees and individual pedigrees. Hence, it suffices to give a (partial) reconstruction $\hat{\mathcal{P}}$ of $\mathcal{P}$ to (partially) reconstruct the original individual pedigree $\mathcal{P}_{\mathsf{indiv}}$. Thus the content of our main result Theorem 3.6.1 and the remainder of this paper primarily work with the coupled pedigree $\mathcal{P}$.

Parameters: For convenience, we collect the various parameters of interest here.

Parameter	**Description**	**Value**
N	Size of founding population	
B	Number of blocks for each individual	$\Theta(\log(N))$
α	Expected # of children per couple	$\Theta(1)$
T	Number of generations in population	$\varepsilon \log(N)$, $\varepsilon = O(1/\log(\alpha))$
$\lvert\Sigma\rvert$	Size of block alphabet	∞

We set $B = O(\log(N))$ for a sufficiently large constant. The expected number of children per couple, α, will be set to a sufficiently large constant that is at least 3. Finally, the number of generations T will be set to $\varepsilon \log(N)$, where $\varepsilon > 0$ is sufficiently small with respect to $1/\log(\alpha)$.

3.4.2 Concentration bounds and upper bounds on inbreeding

In this section we quantify the degree of inbreeding in $\mathcal{P}$. To do so, we first describe an alternative description of our generative model. An equivalent procedure for constructing the coupled pedigree structure $\mathcal{P}$ is to (1) sample the generation sizes according to Poisson random variables with appropriate parameters, (2) pair up individuals in each generation at random into coupled nodes, and (3) have coupled nodes choose two parent coupled nodes at random from the previous generation. This is described formally below.

Lemma 3.4.2. *The (coupled) pedigree $\mathcal{P}$ described in Section 3.4.1 can be equivalently viewed as follows:*

1. *Let $N_T := N$ be the size of the founding population. For i from T to 1: Let $N_i' \overset{def}{=} \lfloor N_i/2 \rfloor \cdot 2$ be the number of individuals in couples, and sample $N_{i-1} \sim \mathsf{Pois}(\alpha N_i'/2)$.*

2. *For each level i, match the individuals at level i randomly, leaving out a single individual if N_i was odd.*

3. *For each level i, sample a vector $\mathbf{v} \in [N_i'/2]^{N_{i-1}}$ from a Multinomial distribution with parameters*
$$(N_{i-1}, (2/N_i', \ldots, 2/N_i')).$$
For any $k \in [N_i'/2]$, the set of coordinates $\{j : v_j = k\}$ are interpreted as children of the k^{th} couple at level i (and are therefore siblings at level $i-1$).

4. *Convert the resulting pedigree on individuals from steps 1–3 to a coupled pedigree $\mathcal{P}$.*

Proof. The number of vertices at each level in the statement of Lemma 3.4.2 is the same as the model in Section 3.4.1. This follows by induction. The number of founding vertices N

is the same in both models. In the model in Section 3.4.1, the number of individuals at level $i-1$ is distributed as $\sum_{j=1}^{N_i'/2} X_j$, where the X_j are iid $\mathsf{Pois}(\alpha)$ and N_i' is the number of individuals at level i that are matched. The value of this sum is distributed as $\mathsf{Pois}(\alpha N_i'/2)$ (due to Proposition 3.3.14), the same as in the statement Lemma 3.4.2.

The random matching in Step 2 of Lemma 3.4.2 is the same as the matching in Section 3.4.1.

The final step in the process above assigns individuals in V_{i-1} to parents in V_i by sampling a vector $\mathbf{v}$ of length N_{i-1} with entries in $[N_i'/2]$ from a multinomial distribution and assigning individuals to parents based on these labels. Indeed, if we look at the number of children of a fixed couple (say, the j^{th} couple in V_i), this is distributed as $\mathsf{Bin}(X, 2/N_i')$, where $X \sim \mathsf{Pois}(\alpha N_i'/2)$. By Poisson thinning (Proposition 3.3.13), this distribution is simply $\mathsf{Pois}(\alpha)$, which is exactly the distribution of the number of children of the j^{th} couple in Section 3.4.1. □

Next we use tail bounds on Poisson random variables to show that the sizes of each level are well-concentrated with high probability, assuming a sufficiently large size of the initial population. Recall that N_i denotes the number of *individuals* in generation i.

Lemma 3.4.3 (Concentration of generations). *Fix δ such that $0 < \delta < \alpha/2 - 1$, and suppose that the founding population size N is at least $\alpha/\delta + 1$. Then, for some constant $C_1 = C_1(\delta)$, with probability at least $1 - T\exp(-C_1\alpha N)$ we have that, for all $i \in \{0, \ldots, T-1\}$*

$$(\alpha/2 - \delta)N_{i+1} \leq N_i \leq (\alpha/2 + \delta) \cdot N_{i+1}. \tag{3.1}$$

Remark 3.4.4. *An immediate corollary of this result is that*

$$(\alpha/2 - \delta)^i \cdot N \leq N_{T-i} \leq (\alpha/2 + \delta)^i \cdot N \tag{3.2}$$

for each $i \leq T$ with high probability.

Proof of Lemma 3.4.3. Our goal is to upper bound the right-hand-side of

$$\mathbb{P}[\text{some } N_j \text{ fails Eq. (3.1)}] \leq \sum_{i=0}^{T-1} \mathbb{P}[N_i \text{ fails Eq. (3.1)} | N_{i+1} \text{ satisfies Eq. (3.2)}]$$

and so it suffices to show

$$\mathbb{P}[N_i \text{ fails Eq. (3.1)} | N_{i+1} \text{ satisfies Eq. (3.2)}] \leq 2\exp(-\Theta(\alpha^2(N-1)/(\alpha+\delta))).$$

Consider fixing the number of individuals at level $i+1$ to be an arbitrary number N_{i+1} satisfying Eq. (3.2). We know that the number of individuals at level i is distributed as $N_i \sim \mathsf{Pois}(\alpha N'_{i+1}/2)$. By applying the Poisson tail bound Proposition 3.3.15, we see that

$$\begin{aligned} &\mathbb{P}\Big[|N_i - \alpha N'_{i+1}/2| > (\delta/2)N'_{i+1} \mid N_{i+1} \text{ satisfies Eq. (3.2)}\Big] && (3.3)\\ &< 2\exp\left(\frac{-(\alpha N'_{i+1}/2)^2}{2(\alpha/2+\delta/2)N'_{i+1}}\right) \\ &< 2\exp\left(-\alpha\frac{-N'_{i+1}}{4(1+\delta)}\right) && (3.4)\end{aligned}$$

We now claim that $|N_i - \alpha N_{i+1}/2| > \delta N_{i+1}$ implies that $|N_i - \alpha N'_{i+1}/2| > (\delta/2)N'_{i+1}$, which follows from the facts that $|N_{i+1} - N'_{i+1}| \leq 1$ and that $N_{i+1} \geq N$ (Eq. (3.2)). Namely, assume that $N_i > (\alpha/2+\delta)N_{i+1}$. Then $N_i > (\alpha/2+\delta/2)N'_{i+1}$, since $N_{i+1} \geq N'_{i+1}$. Now assume instead that $N_i < (\alpha/2-\delta)N_{i+1}$. Then

$$\begin{aligned} N_i &< (\alpha/2-\delta)N_{i+1} \\ &\leq (\alpha/2-\delta)(N'_{i+1}+1) \\ &\leq (\alpha/2-\delta/2)(N'_{i+1}) \end{aligned}$$

where in the last line we use the fact that $(\delta/2)N'_{i+1} \geq (\delta/2)(N-1) \geq \alpha/2$.

Hence, we get that

$$\mathbb{P}[|N_i - \alpha N_{i+1}/2| > \delta N_{i+1} \mid N_{i+1} \text{ satisfies Eq. (3.2)}] \le 2\exp\left(-\alpha\left[\frac{N-1}{4(1+\delta)}\right]\right)$$

where we use the fact that $N_{i+1} \ge N$ since N_{i+1} satisfies Eq. (3.2). □

Remark 3.4.5 (Dependence on δ)**.** *The strategy from this point onwards is to condition on the event from Eq.* (3.1)*. Since this event fails with probability that is exponentially small in N, we lose only an additive* $\exp(-c_\delta \alpha N)$ *probability.*

As mentioned in Section 3.2.1, two nodes may have significantly higher amounts of symbol overlap caused by inbreeding in their ancestral pedigree than would be expected given their distance in the pedigree. This can cause us to reconstruct an incorrect pedigree if we attempt to explain the symbol overlap without accounting for inbreeding; for instance, we may see two nodes and think they are siblings, when in reality they are cousins with inbreeding in their family tree (see Section 3.2.1 for a detailed example). To formally connect different patterns of inbreeding with the amount of spurious symbol overlap they cause, we introduce the notion of *collisions* in an ancestral pedigree. Roughly speaking, triples of coupled nodes with relatively few collisions in their ancestral pedigree do not have many spurious overlaps, which we prove in Section 3.5. We first define collisions and then bound the number that occur under our probabilistic assumptions in Lemma 3.4.8. We also give an alternative characterization of collisions in Lemma 3.4.7 that is useful later.

Definition 3.4.6 (Collisions)**.** *Let $\mathcal{P}$ denote a coupled pedigree. Fix a subset of nodes $A \subset V_k(\mathcal{P})$, where $k \neq T$. If $k > 0$, we say that this collection has z* collisions at level $k+1$ *if the set of parents of A in $\mathcal{P}$ has size $2|A| - z$. If $k = 0$, we say that it has z collisions at level 1 if the set of parents in $\mathcal{P}$ has size $|A| - z$. Write*

$$\mathsf{coll}_{k+1}(A) := (\#\textit{ collisions at level } k+1 \textit{ in } A)$$

Extend the notion of collisions to ancestral subgraphs as follows. If we have nodes $u_1, \ldots, u_J \in$

$V_k(\mathcal{P})$, the number of collisions between the ancestral subpedigrees $\mathsf{anc}(u_j)$ *for* $j = 1,\ldots,J$ *is equal to*

$$\mathsf{coll}(u_1,\ldots,u_J) := \sum_{i=0}^{T-k-1} \mathsf{coll}_{i+1}(\mathsf{anc}_i(u_1) \cup \cdots \cup \mathsf{anc}_i(u_J))$$

where $\mathsf{anc}_i(u_j)$ *denotes the set of ancestors i levels above u_j.*

Lemma 3.4.7 (Ancestral collisions, alternate characterization)**.** *Let $u_1,\ldots,u_J$ denote a set of nodes that are all at the same level. Consider the subpedigree $\mathcal{T} = anc(u_1,\ldots,u_J)$. Let k_j denote the number of nodes in $\mathcal{T}$ that have outdegree j in the subpedigree $\mathcal{T}$. Then*

$$\mathsf{coll}(u_1,\ldots,u_J) = \sum_{j\geq 2}(j-1)k_j.$$

Proof. Let S denote a set of nodes at level i. Let $k_{ij}(S)$ denote the set of parents of S that have outdegree j in the subpedigree $\mathsf{anc}(S)$. Let $\mathsf{coll}_{i+1}(S)$ denote the number of collisions that S has at level $i+1$. Then we claim that

$$\mathsf{coll}_{i+1}(S) = \sum_{j}(j-1)k_{ij}(S). \tag{3.5}$$

This is true by induction on the cardinality of S, as we now demonstrate. We prove this assuming that S is a set of non-extant coupled nodes; the case for extant nodes is extremely similar. The base case $|S| = 1$ follows because the unique node $u \in S$ either has two distinct parents, in which case there are no collisions and each has outdegree 1, or u has a single parent, in which case the number of collisions is 1 and the parent has outdegree 2. In both cases Eq. (3.5) holds.

For the inductive step, suppose that Eq. (3.5) is valid for all S with $|S| \leq s$. Now consider S with $|S| = s+1$. Choose an arbitrary $u \in S$ and consider $S' = S\backslash\{u\}$. Observe that by Definition 3.4.6 and induction:

$$\begin{aligned}\mathsf{coll}_{i+1}(S) &= 2|S| - |par(S)| \\ &= 2|S'| - |par(S')| + 2|\{u\}| - |par(u) \backslash par(S')| \\ &= \mathsf{coll}(S') + 2 - |par(u) \backslash par(S')| \\ &= \sum_j (j-1) k_{ij}(S') + 2 - |par(u) \backslash par(S')|.\end{aligned}$$

Therefore, if u has $\ell \in \{0,1,2\}$ parents contained in $par(S')$, then

$$\mathsf{coll}_{i+1}(S) = \ell + \sum_j (j-1) k_{ij}(S') = \sum_j (j-1) k_{ij}(S),$$

because each parent of u contained in $par(S')$ increases the degree of some node in S' by 1.

Applying this argument over all levels i to the sets $\cup_{\ell=1}^{J} \mathsf{anc}_i(u_\ell)$, we see by Definition 3.4.6 and summing over all levels i that Lemma 3.4.7 holds for coupled nodes. □

In our model and in light of Lemma 3.4.2, a collision between sets A and B intuitively corresponds to a node in B "choosing" a parent couple that was already chosen by another node in $A \cup B$. This observation lets us bound the number of collisions between the ancestors of 3 nodes with high probability.

Lemma 3.4.8 (Exponential tail of collisions). *Fix three nodes $u, v, w \in \mathcal{P}$ in the same level k, and let c be a positive integer. Then*

$$\mathbb{P}[\mathsf{coll}(u,v,w) \geq c] = O\left(\frac{72^c \cdot 2^{2cT}}{N^c}\right) \tag{3.6}$$

Proof. We show that the probability on the left-hand-side of Eq. (3.6) can be upper bounded by the probability that a binomial random variable with sufficiently small mean is at least c, from which the result follows.

We assume that each level has at least N individuals. This is a high probability event by Lemma 3.4.3 (which actually describes a much stronger situation). Since we just want an upper bound, we condition such an event and this assumption is made without loss of generality.

Let $S_i := \mathsf{anc}_i(u) \cup \mathsf{anc}_i(v) \cup \mathsf{anc}_i(w)$. Note that $|S_i| \leq 3 \cdot 2^i$, regardless of how many collisions have happened underneath it. The distribution of $\mathsf{coll}(\mathsf{anc}_i(u), \mathsf{anc}_i(v), \mathsf{anc}_i(w))$ is equal to a sum of at most $3 \cdot 2^{i+1}$ Bernoulli random variables, two for each node in S_i, which are indicator random variables that a parent coupled node selected by some node in $u \in S_i$ is the same as a parent coupled node previously selected by $v \in S_i$ (Lemma 3.4.2). Furthermore, each of these indicator random variables is 1 with probability at most $3 \cdot 2^{T+2}/N$, even conditioned on the previously set random variables—indeed, there are only $3 \cdot 2^{i+1} \leq 3 \cdot 2^T$ parents selected in total, so there are only this many nodes that can be selected from to cause a collision, and there are at least $\lfloor N/2 \rfloor \geq N/4$ coupled nodes at level $i+1$. Therefore, the random variable $\mathsf{coll}(S_i)$ is stochastically dominated by $\mathsf{Bin}(3 \cdot 2^{i+1}, 3 \cdot 2^{T+2}/N)$. Let $X_i \sim \mathsf{Bin}(3 \cdot 2^{i+1}, 3 \cdot 2^{T+2}/N)$. Then we get that

$$\begin{aligned}
\mathbb{P}[\mathsf{coll}(u,v,w) \geq c] &= \mathbb{P}[\sum_i \mathsf{coll}_{k+i}(S_i) \geq c] \\
&\leq \mathbb{P}[\sum_{i=k}^{T-1} X_i \geq c] \\
&\leq \mathbb{P}[X \geq c] \qquad (3.7)
\end{aligned}$$

where $X \sim \mathsf{Bin}(3 \cdot 2^{T+1}, 3 \cdot 2^{T+2}/N)$. By bounding the binomial tail using Eq. (1.1) and noting that we take $N > 144 \cdot 2^{2T}$, (Eq. (3.7)) can be bounded by

$$\begin{aligned}
\mathbb{P}[X \geq c] &\leq \sum_{i=c}^{3 \cdot 2^{T+1}} \binom{3 \cdot 2^{T+1}}{i} \left(\frac{3 \cdot 2^{T+2}}{N} \right)^i \\
&\leq \sum_{i=c}^{3 \cdot 2^{T+1}} (3 \cdot 2^{T+1})^i \left(\frac{3 \cdot 2^{T+2}}{N} \right)^i \\
&\leq 2 \cdot 72^c \cdot \frac{2^{2cT}}{N^c}
\end{aligned}$$

□

In particular, by union bounding over all triples of nodes in the coupled pedigree $\mathcal{P}$, we get the following corollary. Note that there are most $(\alpha/2+\delta)^T \cdot N$ nodes in the pedigree when we condition on the high-probability event from Lemma 3.4.3.

Corollary 3.4.9.

$$\mathbb{P}[\exists u,v,w : \mathsf{coll}(u,v,w) \geq 4] = O\left(\frac{(\alpha/2+\delta)^{3T} 2^{8T}}{N}\right)$$

Since we take the ratio $T/\log(N)$ to be sufficiently small (Section 3.4.1), the probability of the above event is negligible. Hence, we can assume without loss of generality for the rest of the document that the number of collisions in the ancestral trees of any three nodes is at most 3.

Additionally, by applying Lemma 3.4.8 to a single node (repeated three times) and applying linearity of expectation, we can bound the probability that there are many coupled nodes u with collisions in their ancestral pedigrees $\mathsf{anc}(u)$ using Markov's inequality. We state this as a corollary.

Corollary 3.4.10. *For any $C > 0$,*

$$\mathbb{P}\left[\left|\{u : \mathsf{coll}(u) \geq 1\}\right| \geq C(2\alpha + 4\delta)^T\right] \leq 72/C$$

as long as N is sufficiently large.

Definition 3.4.11 (d-Richness)**.** *Fix a pedigree $\mathcal{P}$, and let $d \geq 3$ be an integer. All extant nodes in $\mathcal{P}$ are d-rich. For all $k > 0$, a level k-node is d-**rich** if it has at least d children that are d-rich.*

Lemma 3.4.12 (Most nodes are d-rich)**.** *Fix a constant $0 < \tau < 1$, and let $\delta > 0$ as in Lemma 3.4.3. As long as N and α are sufficiently large, there exists a constant $C_2 = C_2(\tau, \delta)$ such that with probability $1 - T\exp(-C_2\alpha N)$, at least $(1-\tau)$ fraction of level-k coupled nodes in $\mathcal{P}$ are d-rich for all k.*

Proof of Lemma 3.4.12. Let the term "d-poor node" refer to coupled nodes that are not d-rich. Let M_k denote the number of coupled nodes at level k in $\mathcal{P}$. Our goal is to prove an upper bound on the event that there are at least τM_{k+1} d-poor nodes at level $k+1$, conditioned on the event that there are at least $(1-\tau)M_k$ d-rich nodes at level k.

Let R_k denote the event that there are at least $(1-\tau)M_k$ d-rich nodes at level k. Let E denote the event $(\alpha/2-\delta)M_{k+1} \le M_k \le (\alpha/2+\delta)M_{k+1}$ for all k, which occurs with probability $1-\exp(-C_1\alpha N)$ by Lemma 3.4.3. We also condition on the sizes of $M_0, \dots, M_T$, abbreviating this conditioning as $M_{0:T}$.

Let S be an arbitrary subset of nodes at level $k+1$ of size $\tau M_{k+1}+1$, and consider the event where S only consists of d-poor nodes. This implies that the number of d-rich children of S is at most $(d-1)(\tau M_{k+1}+1)$. Let X_i be iid Bernoulli RVs, which represent indicators for the event where the ith d-rich child chooses at least one of its parents to be in S. Note that $\mathbb{P}(X_i = 1) = \left(1 - \left(1 - \frac{|S|}{M_{k+1}}\right)^2\right) > \frac{|S|}{M_{k+1}}$.

$$\begin{aligned}
&\mathbb{P}(S \text{ only has } d\text{-poor nodes} \mid R_k, E, M_{0:T}) \\
&\quad \le \mathbb{P}\left[\sum_{i=1}^{(1-\tau)M_k} X_i \le (d-1)|S| \,\middle|\, M_{0:T}\right] \\
&\quad \le \exp\left[-\frac{(1-\tau)M_k|S|}{2M_{k+1}}\left(1 - \frac{(d-1)M_{k+1}}{(1-\tau)M_k}\right)^2\right] \qquad \textit{(Chernoff–Hoeffding Bound)}
\end{aligned}$$

Observe that there are $\binom{M_{k+1}}{|S|} \le \left(\frac{e}{\tau}\right)^{\tau M_{k+1}+1}$ many choices for S (Eq. (1.1)). To apply a union bound, it suffices for α to be large enough so that $\frac{(1-\tau)M_k}{M_{k+1}}\left(1 - \frac{(d-1)M_{k+1}}{(1-\tau)M_k}\right)^2 \approx (1-\tau)\alpha(1-\frac{d-1}{(1-\tau)\alpha})^2$ looks linear in α. In that case, we obtain a bound of the form

$$\begin{aligned}
&\mathbb{P}(\text{at least } \tau M_{k+1} \ d\text{-poor nodes at level } k+1 \mid R_k, E, M_{0:T}) \\
&\qquad \le \exp(-CM_{k+1}\alpha).
\end{aligned}$$

Therefore, we may write

$$\begin{aligned}
&\mathbb{P}(\text{at least } (1-\tau) \text{ fraction of } d\text{-rich at all levels}) \\
&\qquad \geq (1-e^{-C_1\alpha N})\prod_{k=1}^{T}(1-\exp(-CM_{k+1}\alpha)) \\
&\qquad \geq 1-\exp(-C_1\alpha N)-\sum_{k=0}^{T-1}\exp(-CN(\alpha/2-\delta)^k\alpha) \\
&\qquad \geq 1-T\exp(-C_2\alpha N)
\end{aligned}$$

for an appropriate constant C_2 depending only on τ and δ.

□

Lemma 3.4.13 (Cliques have unique parents)**.** *Let G_k denote the siblinghood hypergraph at level k. Let $\delta > 0$ be as in Lemma 3.4.3. For a constant $C_3 = C_3(\delta)$, with probability at least $1 - \frac{1}{N}e^{C_3 T \log \alpha}$, for all hypercliques $\mathcal{C} \subset G_k$ with at least one hyperedge, there is a unique node at level $k+1$ that is a parent of every node in $\mathcal{C}$. We refer this node as the* ***parent*** *of $\mathcal{C}$.*

Proof. By Proposition 3.3.12, a hyperclique corresponds to a set of coupled nodes that contain a set of mutual siblings, where each couple has at least one of the siblings in it. This establishes that there is a coupled node at level $k+1$ that is at least one parent of every node in $\mathcal{C}$. In the case where $\mathcal{C}$ is a hyperclique of extant nodes, we are done: every node in $\mathcal{C}$ is an individual and has exactly one parent coupled node.

If $\mathcal{C}$ is at a higher level, note that there can be at most two parents for $\mathcal{C}$, as defined above. The reason is that any individual has exactly one parent couple, and since there are only two individuals in a couple, there cannot be three parent couples each with one child in each couple in $\mathcal{C}$.

Next we show that if there are two coupled nodes, both of which are parents of $\mathcal{C}$, then there must be many collisions among the ancestors of $\mathcal{C}$, and therefore we can rule this out as a low-probability event. Since $\mathcal{C}$ has at least one hyperedge, we know that $|\mathcal{C}| \geq 3$. This means that any arbitrary set of three nodes from $\mathcal{C}$ must have at least $6-2=4$ collisions by Definition 3.4.6—but Corollary 3.4.9 shows that with probability $O\left(\frac{(\alpha/2+\delta)^{3T}2^{8T}}{N}\right)$, this does not occur anywhere in the pedigree. □

Lemma 3.4.14 (Disjointness of maximal cliques). *Let G_k denote the siblinghood hypergraph at level k. For $k = 0$, each extant node is contained in a unique maximal clique, and moreover, the maximal cliques in G_0 are vertex disjoint (and thus, also edge-disjoint). For $k > 0$, each node is contained in at most two maximal cliques. Moreover, with probability $1 - \frac{1}{N}e^{C_3 T \log \alpha}$, the maximal cliques in G_k are edge-disjoint.*

Proof. Note that maximal cliques in the siblinghood hypergraph correspond to maximal sets of siblings. The claim for extant nodes is relatively trivial - extants are individuals, and so the maximal sets of siblings partition the set of extant nodes.

For $k > 0$, since each individual in a coupled node has one pair of parents, a coupled node can have at most two parents. Thus it can be part of at most two sets of siblings. Hence, it is part of at most two maximal cliques.

Finally, we need to establish that the maximal cliques in G_k are edge-disjoint. To do this, it suffices to show that the intersection between any two maximal cliques is less than 3, so there can be no hyper-edge. Indeed, if three nodes that are simultaneously in two maximal cliques, these three nodes would themselves form a clique with two different parents in level $k+1$, which occurs with probability at most $1 - \frac{1}{N}e^{C_3 T \log \alpha}$ by Lemma 3.4.13. □

3.4.3 The joint LCA and its uniqueness

The next two lemmas are crucial in Section 3.6 to show that we can accurately collect symbols for accurately reconstructed coupled nodes. Here we define the *joint lowest common ancestor*, which is a special type of LCA for a triple of coupled nodes.

Definition 3.4.15. *Let u, v, w denote coupled nodes in $\mathcal{P}$. We say that u, v, w have a **joint LCA** z if it holds that $z \in \mathsf{LCA}(u, v, w)$ and there exist distinct children c_u, c_v, c_w of z so that for all $x \in \{u, v, w\}$, c_x is an ancestor of x.*

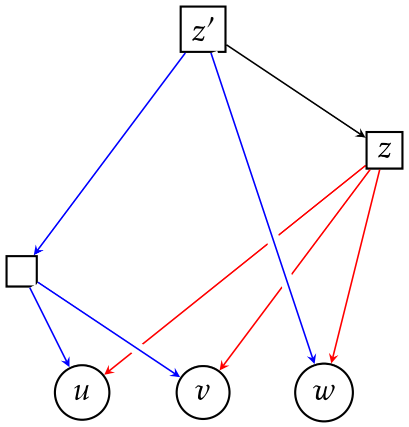


Figure 3-5: *"Proof-by-picture" of Lemma 3.4.17.*

Lemma 3.4.16 (Joint LCA is unique)**.** *Suppose that each triple of coupled nodes in $\mathcal{P}$ has at most 3 collisions. Further suppose that u, v, w have a joint LCA $z \in \mathsf{LCA}(u, v, w)$. Then z is the unique LCA of u, v, w.*

Proof. For the sake of contradiction, suppose that u, v, w have another LCA $z' \neq z$. By the definition of LCA, z' is neither an ancestor nor a descendant of z.

If z' is a joint LCA of u, v, w, then both z and z' have outdegree 3 in $\mathsf{anc}(u, v, w)$, which by Lemma 3.4.7 implies that $\mathsf{anc}(u, v, w)$ has at least $2 \times (3 - 1) = 4$ collisions.

If z' is not a joint LCA, then z' has outdegree 2 in $\mathsf{anc}(u, v, w)$. Moreover, there exists a unique lowest node $y \in \mathsf{desc}(z') \cap \mathsf{anc}(u, v, w)$ that is an ancestor of precisely two nodes in $\{u, v, w\}$. In particular, y has outdegree at least 2 in $\mathsf{anc}(u, v, w)$. Observe that the nodes y, z, z' are all distinct. Hence by Lemma 3.4.7, the number of collisions is at least $2 \times (2 - 1) + 1 \times (3 - 1) = 4$.

In either case, $\mathsf{anc}(u, v, w)$ has at least 4 collisions, which is a contradiction. □

Lemma 3.4.17 (Inheritance paths go through LCA)**.** *Suppose that each triple of coupled nodes in $\mathcal{P}$ has at most 3 collisions. Further suppose that $u, v, w \in \mathcal{P}$ have an LCA z. Let z' denote a strict ancestor of z. Then for some $x \in \{u, v, w\}$, all paths from z' to x in $\mathcal{P}$ pass through z.*

Proof. To draw a contradiction, suppose that for all $x \in \{u, v, w\}$ that z' has a path to x that does not go through z. Suppose further, without loss of generality, that z' is the lowest node in $\mathcal{P}$ that is an ancestor of z and has this property.

Let $\mathcal{T}$ denote a spanning tree on $\mathsf{desc}(z) \cap \mathsf{anc}(u,v,w)$ (red edges in Fig. 3-5). Also select a spanning tree $\mathcal{T}'$ on the union of all paths from z' to u,v,w that do not go through z (blue edges in Fig. 3-5). Observe that z' has outdegree at least 2 in $\mathcal{T}'$. Since z' also has a path to z, then z' has outdegree at least 3 in $\mathsf{anc}(u,v,w)$. Moreover, $\mathcal{T}$ has 2 collisions. Since z' is not contained in $\mathcal{T}$, we conclude by Lemma 3.4.7 that $\mathsf{anc}(u,v,w)$ has at least $2+1\times(3-1)=4$ collisions. The first terms accounts for the collisions in $\mathcal{T}$, and the second applies Lemma 3.4.7 to z'. This is a contradiction. □

Note that by Corollary 3.4.9, Lemmas 3.4.16 and 3.4.17 hold for all triples $u,v,w \in \mathcal{P}$ with high probability.

3.5 Lemmas that enable reconstruction

In this section, we prove bounds on "overlap statistics" previously explored in Section 3.2. Since we now have switched to talking about coupled pedigrees, we re-define its notion now.

Definition 3.5.1 (Diploid blocks)**.** *Let $\mathcal{P}_{\mathsf{indiv}}$ induce the coupled pedigree $\mathcal{P}$. Given (haploid) gene sequences $(\sigma_u)_{u \in V(\mathcal{P}_{\mathsf{indiv}})}$, we associate with each non-extant couple $v = \{v_1, v_2\}$ node a* ***diploid sequence*** *σ_v defined in terms of each block b as a multiset $\sigma_v(b) := \sigma_{v_1}(b) \cup \sigma_{v_2}(b)$. Each extant node's block is thought of as a singleton set.*

Definition 3.5.2 (Diploid overlap)**.** *Three diploid sequences $\sigma, \sigma', \sigma''$* ***overlap*** *in block b if*

$$\sigma(b) \cap \sigma'(b) \cap \sigma''(b) \neq \varnothing.$$

The term ***fraction of mutual overlaps*** *between coupled nodes u,v,w in refers to the statistic*

$$\frac{\#\text{ overlapping blocks of } \sigma_u, \sigma_v, \sigma_w}{B} = \frac{|\{b \in [B] : \sigma_u(b) \cap \sigma_v(b) \cap \sigma_w(b) \neq \varnothing\}|}{B}.$$

3.5.1 Distinguishing siblings from non-siblings: Coincidence probability bounds

In this section, we establish the following high-probability separation condition for triples of coupled nodes at the same level:

- if u, v, w are mutually siblings, they overlap in at least 1/4 fraction of blocks.
- if u, v, w are not mutually siblings, they overlap in at most 3/16 fraction of blocks.

In order to reconstruct the pedigree, we perform inference on the underlying pedigree structure from the symbols at the extant level. The key step of our reconstruction algorithm is to infer which triples of nodes are mutually siblings based on the overlap between their reconstructed symbols. The conditions stated above justify using the number of overlapping symbols in triples as a statistic for determining siblinghood. The first fact (Lemma 3.5.3) is easy to prove. In contrast, the second fact (Lemma 3.5.4) is rather non-trivial; we prove it using casework.

Lemma 3.5.3 (Symbol overlap in siblings)**.** *With probability* $1 - O(\alpha^{3T} N^3 \exp(-\gamma^2 B))$*, the fraction of mutual overlap in symbols between any triple of coupled nodes* u*,* $v, w \in \mathcal{P}$ *that are mutually siblings is at least* $\frac{1}{4} - \gamma$ *for any arbitrarily small* $\gamma > 0$*.*

Proof. It suffices to consider the overlap of the individuals u_1, v_1, w_1 in u, v, w, respectively, that are siblings, *i.e.*, u_1, v_1, w_1 have a common parent in $\mathcal{P}_{\mathsf{indiv}}$. We claim that the expected fraction of overlap for u_1, v_1, w_1 is at least $1/4$. Indeed, any individual symbol at the parent (couple) node survives to all three children with probability $1/8$, and there are $2B$ symbols at the parent (one per block per member of the couple). The Chernoff–Hoeffding bound gives that for any fixed triple (u, v, w) of siblings, the probability that it has less than $1/4 - \gamma$ mutual overlap is at most $\exp(-\gamma^2 B)$. To be explicit, let X_i denote the indicator of an overlap between u, v, w in block b.

$$\begin{aligned}\mathbb{P}(\text{average overlap} < 1/4 - \gamma) &= \mathbb{P}\left(\frac{1}{B}\sum_{i=1}^{B} X_i < 1/4 + \gamma\right)\\ &= \mathbb{P}\left(\frac{1}{B}\sum_{i=1}^{B}(X_i - \mathbb{E}[X_i]) < 1/4 - \mathbb{E}[X_1] + \gamma\right)\\ &\leq \mathbb{P}\left(\frac{1}{B}\sum_{i=1}^{B}(X_i - \mathbb{E}[X_i]) < -\gamma\right)\\ &\leq 2\exp(-2B\gamma^2).\end{aligned}$$

In the second line we use that X_i are i.i.d., in the third line we use that the expectation is at least 1/4, and to finish we apply Chernoff–Hoeffding. A union bound over all $O((\alpha^T N)^3)$ triples of siblings yields the result. □

Lemma 3.5.4 (Symbol overlap in non-siblings). *Fix $\gamma > 0$. With probability $1 - O(1/N_T) - O(\alpha^{3T}N^3\exp(-\gamma^2 B))$, every triple of coupled nodes u, v, and w that are at the same level but are **not** mutual siblings share overlap in less than $\frac{3}{16} + \gamma$ fraction of their symbols.*

Proof of Lemma 3.5.4

Remark 3.5.5. *In this proof, we condition on the high probability event from Corollary 3.4.9 that all triples u, v, w of coupled nodes have at most 3 collisions in their ancestral subpedigree* $\mathsf{anc}(u,v,w)$.

It is clear that if u, v, w are completely unrelated, then their mutual overlap is zero, since we assume an infinite alphabet. If u, v, w have a common ancestor, then typically their ancestral pedigree has two collisions, and all triples have at most three collisions in their ancestral pedigree by our conditioning in Remark 3.5.5. We refer to triples with three collisions as being *inbred* and think of the extra collision as the *site* of inbreeding, a notion that we later formalize in this section.

Recall the definition of tree subpedigree (Definition 3.3.9), which we refer to simply as a *tree* in what follows. Also recall that an edge of multiplicity 2 in a pedigree is considered to be an undirected cycle of length 2. Thus, a tree subpedigree consists only of simple (multiplicity 1) edges. Our strategy for proving Lemma 3.5.4 follows the recipe below for casework.

1. u,v,w have exactly two LCAs, and the ancestral pedigree of u,v,w is a tree.
2. u,v,w have exactly one LCA, and the LCA has a cycle above it.
3. u,v,w have exactly one LCA, and the ancestral pedigree of u,v,w is a tree.
4. u,v,w have exactly one LCA, and the ancestral pedigree of u,v,w contains a cycle that is not completely above the LCA.

We now assert that the above cases cover all possibilities; this is proven in the next two claims.

Claim 3.5.6. *For u, v, and w to have a single LCA, their ancestors must have at least 2 collisions.*

Proof. All three nodes need a common ancestor, which means there are at least 2 collisions are present in $\mathsf{anc}(u,v,w)$. □

Claim 3.5.7. *The nodes u, v, and w have at most two LCAs, with two LCAs only if $\mathsf{anc}(u,v,w)$ has three collisions. Furthermore, if there are two LCAs, then $\mathsf{anc}(u,v,w)$ is a tree pedigree.*

Proof. By the previous claim, creating a single LCA for three nodes requires 2 collisions in $\mathsf{anc}(u,v,w)$. By definition, one LCA cannot be an ancestor of another LCA. This means there must be at least one more collision in $\mathsf{anc}(u,v,w)$ to create the second LCA, bringing the total number of collisions required in $\mathsf{anc}(u,v,w)$ to three. This immediately yields the final part of the claim by Remark 3.5.5.

To establish that there are at most two LCAs, suppose we add a third LCA. Then by the same argument, this LCA cannot be an ancestor of either of the two other LCAs, and so there must be another collision to explain it. This leads to four collisions among the ancestors, which we have ruled out. □

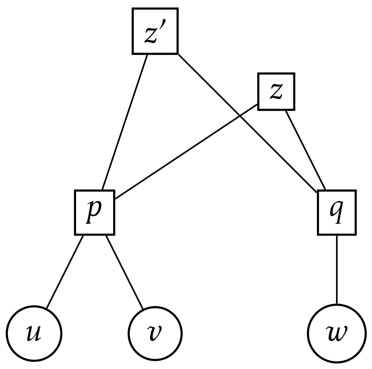


Figure 3-6: *The topologies of Lemma 3.5.8 with two LCAs. Others are obtained by swapping the roles of u, v, w.*

We now upper bound the expected overlap between u, v and w by doing the above casework on the structure of their ancestral pedigrees. We simply upper bound the expected overlap, relying on the independence of inheritance in the different blocks so that we can apply a Chernoff–Hoeffding bound.

Lemma 3.5.8 (Case 1: exactly two LCAs)**.** *Suppose that u, v, and w have exactly two LCAs. Then the expected fraction of mutual overlap is at most* 1/8*.*

Proof. Fig. 3-6 illustrates the topology of interest. First we note that neither of the LCAs can have repeated symbols, since their ancestral pedigrees contain no collisions. Consider the ancestral pedigree from u, v, and w up to any one particular LCA, noting that this pedigree is a tree by Claim 3.5.7. Any configuration containing u, v, w and their ancestors leading up to that LCA has at least 5 edges, since u, v, w are not mutual siblings. Therefore, the probability that a single symbol propagates from that LCA to all of u, v, and w is $\leq (1/2)^5 = 1/32$, which yields an expected $1/16$ fraction of overlap since there are $2|B|$ symbols at the LCA (since it is a coupled node). Since there are two such LCAs, the expectation is at most 1/8. □

In the remaining cases, we assume there is exactly one LCA. Note that any common symbols across u, v, and w must be present in this LCA—if u, v, and w inherit a symbol that is not present in this LCA, then by tracing their paths of inheritance for the symbol we can find another LCA. However, this does not guarantee that *all* common symbols in u,

v, and w can be traced back to inheritance from the LCA— if there is inbreeding, some nodes in $\{u,v,w\}$ can potentially inherit a symbol via an ancestor of the LCA through a path does not go through the LCA, while the rest inherit it from the LCA.

Lemma 3.5.9 (Case 2: one LCA with cycle above)**.** *Suppose that u, v, and w have exactly one LCA z. Furthermore, this LCA has at least one collision in its ancestral pedigree. Then the fraction of mutual overlap is at most* $1/8$ *in expectation.*

Proof. We know that u, v, and w, must have at least two distinct parents between them that are connected to z (else z would be their parent). This means there are at least two edges in the graph between z and the parents of u, v, and w, and at least three edges between u, v, and w and their respective parents.

Since we know there are at most three collisions among the ancestors of u, v, and w, there can be only one collision in the ancestral pedigree of z, and the presence of this collision means there are no other collisions in $\mathsf{anc}(u,v,w)$. Therefore, each of the parent couples of u, v, and w have an individual that is unrelated to z, and so there are no repeated symbols within any of the parent couples. So even if the parents were to get 100% overlap in the blocks due to inheritance from z, it holds that u, v, and w inherit at most $1/8$ fraction of these blocks on expectation.

Finally, all common symbols between u, v, and w must have been inherited from z— if a common symbol was instead inherited by some $x \in \{u,v,w\}$ from some ancestor of z, this would create a fourth collision in $\mathsf{anc}(u,v,w)$. □

Lemma 3.5.10 (Case 3: one LCA and $\mathsf{anc}(u,v,w)$ is a tree)**.** *Suppose u, v, and w have exactly one LCA and $\mathsf{anc}(u,v,w)$ is a tree. Then the fraction of mutual overlap is at most* $1/16$ *in expectation.*

Proof. The lack of any cycles in $\mathsf{anc}(u,v,w)$ means that all inheritance of common symbols comes from the lone LCA z. Any such union of paths from z to u, v and w forms a directed tree with at least five edges; see Fig. 3-7. In addition, z has two distinct symbols in every

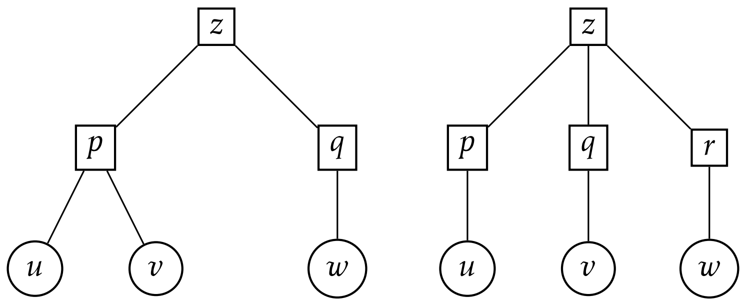


Figure 3-7: *Exhaustive list of topologies from Lemma 3.5.10, up to re-labelling of u, v, w. Each edge represents a path of length > 1.*

block. Therefore, for any particular symbol the probability that all three of u, v, w inherit it is $\leq (1/2)^5 = 1/32$, which yields an expected fraction of at most $1/16$ overlapping blocks. □

The final case is the most complicated one to analyze.

Lemma 3.5.11 (Case 4: one LCA with cycle not completely above)**.** *Suppose u, v, and w have exactly one LCA and* $\mathsf{anc}(u, v, w)$ *contains a cycle that does not lie completely above* $z = \mathsf{LCA}(u, v, w)$*. Then the fraction of mutual overlap is at most* $3/16$ *in expectation.*

As an aid in proving Lemma 3.5.11, it is helpful to first identify the **"most recent" inbred** node. We make this notion precise now.

Definition 3.5.12 (Witness)**.** *We call a node $g \in$* $\mathsf{anc}(u, v, w)$ *a* witness to inbreeding *or simply a* witness *if g is the lowest node in* $\mathsf{anc}(u, v, w)$ *that is part of an undirected cycle.*

Lemma 3.5.13 (Unique witness)**.** *Under the conditions of Lemma 3.5.11, there exists a unique witness in* $\mathsf{anc}(u, v, w)$*. Moreover, this witness lies strictly below the LCA z.*

Proof. We know that $\mathcal{T} := \mathsf{anc}(u, v, w)$ is not a tree, so there exists a cycle in $\mathcal{T}$. We show that there can only be one cycle. Suppose that there exist two cycles $\mathcal{C}, \mathcal{C}'$ in $\mathcal{T}$. Then we claim that $\mathsf{coll}(u, v, w) \geq 4$.

Consider a spanning tree $\mathcal{T}'$ of $\mathcal{T}$. Then $\mathcal{T}'$ has two collisions. Moreover, $\mathcal{T}' \cup \mathcal{C}$ contains a single cycle, so we conclude that there exists a node in $\mathcal{T}'$ whose outdegree is increased by one upon adding the edges from $\mathcal{C}$ to $\mathcal{T}'$ (Otherwise, $\mathcal{T}' \cup \mathcal{C}$ would still be a tree). Therefore, by Lemma 3.4.7, $\mathcal{T}' \cup \mathcal{C}$ has three collisions. By similar reasoning and using that $\mathcal{C} \neq \mathcal{C}'$, we conclude that $\mathcal{T}' \cup \mathcal{C} \cup \mathcal{C}'$ has 4 collisions. Since $\mathcal{T}' \cup \mathcal{C} \cup \mathcal{C}' \subset \mathcal{T}$, we conclude that $\mathcal{T}$ has at least 4 collisions. But under our conditioning, no subpedigree has 4 or more collisions. It follows that in Lemma 3.5.11 there is exactly one cycle in $\mathcal{T}$, and thus, exactly one witness.

To prove the final statement, note that if the witness is located above z in $\mathsf{anc}(u,v,w)$, then the cycle lies completely above z. $\square$

Proof of Lemma 3.5.11. Consider u,v,w and the subpedigree $\mathcal{T} = \mathsf{anc}(u,v,w)$ consisting of the ancestors of u,v,w. Recall that z is the unique LCA of u,v,w. By Lemma 3.5.13, there is a unique witness $g \in \mathcal{T}$, which is the lowest node in the unique cycle occurring in $\mathcal{T}$.

Subcase 1: $\mathsf{LCA}(u,v) = \mathsf{LCA}(v,w) = \mathsf{LCA}(u,w) = \mathsf{LCA}(u,v,w)$.

Without further loss of generality, suppose that the witness g lies along the path from u to z. Then it follows that there is a unique path from v to z in $\mathcal{T}$. Otherwise, there would exist two cycles in $\mathcal{T}$, which is a contradiction as this would lead to 4 collisions in $\mathcal{T}$. Similarly, there is a unique path from w to z in $\mathcal{T}$. Moreover, $\mathsf{anc}(z)$ is a tree. It follows that the subpedigree $\mathsf{anc}(v,w)$ of the ancestors of v and w is a tree. Observe that z is at least two levels above v,w, and by the topology of this subcase, there are at least 4 edges in the tree subpedigree from z to v and w. This implies that the expected overlap between v and w is at most $2 \cdot (1/2)^4 = 1/8$. Thus the expected overlap between u,v,w is at most the expected overlap between u and v, which is bounded by $1/8$.

Subcase 2: Without loss of generality, $\mathsf{LCA}(u,v) \neq \mathsf{LCA}(u,v,w)$.

Let $p = \mathsf{LCA}(u,v)$. Either g is on the branch that leads to u and v, or it is on the branch that leads to w. First, suppose that g is on the branch that leads to u and v. Then we may further assume g is on the path from z to p. For if, say, g is on the path from p to u, then $\mathsf{anc}(v,w)$ is a tree, in which case we can argue as in Subcase 1 that the mutual expected overlap between u,v,w is at most $1/8$.

Therefore, it suffices to consider the cases g is on the path from z to p or g is on the path from z to w (Fig. 3-8). In the first case, the descendants of g form a tree with at least two

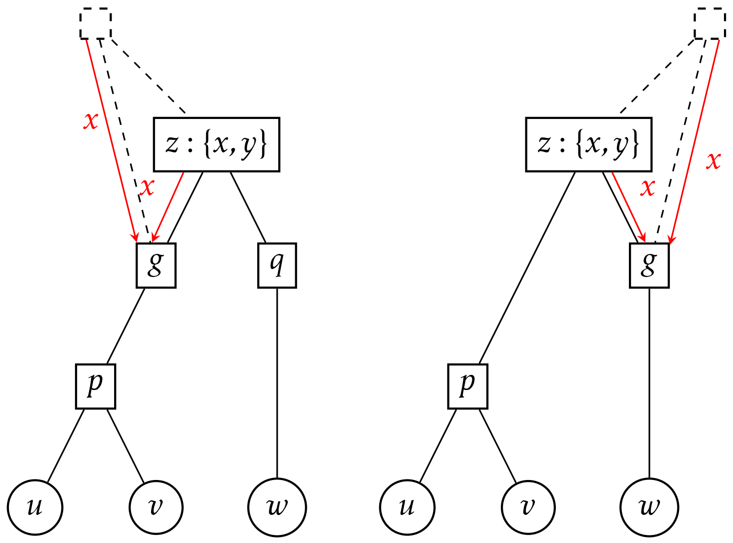


Figure 3-8: *Example of structures being analyzed in the proof of Lemma 3.5.11, Subcase 2. Here $\{x,y\}$ depict the symbols of the LCA z in a specific block. The red edges delineate the inheritance events (possibly occurring simultaneously) of a common symbol x.*

edges. Moreover, there is a unique node q at the same level as g in $\mathcal{T}$, and this individual is located on the path from z to w. Let $\sigma(z) = \{x,y\}$ denote the (distinct) symbols of z in a given block. By these facts, symmetry, and conditional independence of inheritance,

$$\begin{aligned}
&\mathbb{P}[\sigma(u)\cap\sigma(v)\cap\sigma(w)\neq\varnothing]\\
&\leq 2\mathbb{P}[\sigma(g)=\{x,x\},\, x\in\sigma(u)\cap\sigma(v)]\mathbb{P}[x\in\sigma(q), x\in\sigma(w)]\\
&\quad + 2\mathbb{P}[\sigma(g)=\{x,y\}, x\in\sigma(u)\cap\sigma(v)]\mathbb{P}[x\in\sigma(q), x\in\sigma(w)]\\
&\leq 2\times\left(\frac{1}{4}\times 1\right)\times\left(\frac{1}{2}\times\frac{1}{2}\right)+2\times\left(\frac{1}{2}\times\frac{1}{4}\right)\times\left(\frac{1}{2}\times\frac{1}{2}\right)\\
&=\frac{3}{16}.
\end{aligned}$$

The second line includes a factor of 2 to account for either x or y being passed down to u,v,w. The terms in the third line are ordered to correspond to the events in the two lines above. In particular, we have by conditional independence of inheritance that

$$\mathbb{P}[\sigma(g)=\{x,x\}]\leq 1/4$$

because there are at most 2 paths from z to g, and each has probability at most 1/2 of passing down x. The bound

$$\mathbb{P}[\sigma(g) = \{x, y\}] \leq 1/2$$

holds similarly.

Now suppose that g is on the path from z to w. Then

$$\begin{aligned}\mathbb{P}[\sigma(u) \cap \sigma(v) \cap \sigma(w) \neq \varnothing] &\leq 2\mathbb{P}[x \in \sigma(u) \cap \sigma(v)]\mathbb{P}[x \in \sigma(w)] \\ &\leq 2 \cdot \frac{1}{8} \cdot \frac{3}{4} = \frac{3}{16}.\end{aligned}$$

Above, we used the fact that tree pedigree from z to u, v has at least 3 edges. We also used the fact

$$\mathbb{P}[x \in \sigma(w)] \leq \frac{3}{4},$$

which holds because there are at most two paths to w from z, each path has probability at least 1/2 of not passing down x, and so by conditional independence of inheritance, the probability that both paths do not pass down x is at least 1/4. □

Finally, to finish the proof of Lemma 3.5.4 using Lemmas 3.5.8, 3.5.9, 3.5.10, and 3.5.11, note that in all four cases the expected overlap between coupled nodes u, v, w is at most 3/16. Thus, the probability that u, v, w mutually share more than $3/16 + \gamma$ fraction of symbols in all cases is at most $2\exp(-2B\gamma^2)$ by Chernoff–Hoeffding, similar to the analysis of Lemma 3.5.3. Union bounding over all $O((\alpha^T N)^3)$ possible triples gives an $O(\alpha^{3T} N^3 \exp(-B\gamma^2))$ upper bound of the chance that there is some triple with at least $3/16 + \gamma$ overlap. By also ruling out the bad event in Corollary 3.4.9 (which occurs with probability $O(1/N_T)$), we obtain the desired upper bound.

3.5.2 Which ancestors are reconstructible?

In this section, we characterize nodes that are of importance in our analysis: couples whose history *lacks inbreeding (e.g. graph structure is reconstructible using blocks)* and *have ample* *extant information (e.g. blocks are recoverable)*. We present this in two parts respectively in Definition 3.5.14 and Definition 3.5.15.

Definition 3.5.14 (Awesome Node)**.** *Call a node in the pedigree $\mathcal{P}$* awesome *if:*

1. *It is d-rich.*

2. *It is not an ancestor of any extant node that has a collision within its own ancestral pedigree (including itself).*

Definition 3.5.15 (b-goodness)**.** *Let $b \in [B]$ be a specific block. Say that a coupled node v in a pedigree $\mathcal{P}$ is b-**good** if v has at least two sets of three extant descendants x_1, y_1, z_1 and x_2, y_2, z_2 in $\mathcal{P}$ such that:*

1. *v is a joint LCA of x_1, y_1, z_1 and is a joint LCA of x_2, y_2, z_2.*

2. *x_1, y_1, and z_1 all have the same symbol σ_1 in block b, and x_2, y_2, and z_2 all have the same symbol σ_2 in block b.*

3. *$\sigma_1 \neq \sigma_2$.*

We furthermore define every extant node to be b-good, for all $b \in [B]$.

We now deliver the main message of this section: *most nodes have these properties*, given the assumptions of our model (Proposition 3.5.16 and Lemma 3.5.17). Therefore, this characterization enables a natural reconstruction algorithm (Section 3.6).

Proposition 3.5.16 (Many awesome nodes)**.** *Let $d > 0$ (as in Definition 3.5.14) be a constant, let α be a sufficiently large constant with respect to d, and let N be sufficiently large with respect to both d and α. With probability at least $1 - \alpha^{-\Omega(T)}$, in every layer of the pedigree at least $1 - 1/d$ fraction of the nodes are awesome.*

Proof. Since α and N are sufficiently large with respect to d, we can apply Lemma 3.4.12 with $\tau = 1/(2d)$ and $\delta = d$. This tells us that at least $1 - 1/(2d)$ fraction of nodes in each layer are d-rich with probability $1 - T\exp(-C_2\alpha N)$, where the constant $C_2 = C_2(1/(2d), d)$ depends only on d.

Applying Corollary 3.4.10 with $C = \alpha^T$, there are at most $\alpha^{O(T)}$ nodes at the extant level with collisions in their ancestral pedigree, with probability $1 - \alpha^{-\Omega(T)}$. This means there are at most $2^T \cdot \alpha^{O(T)}$ *ancestors* of these nodes. It follows that the number of nodes that are d-rich but not awesome is at most $2^T \cdot \alpha^{O(T)}$. This is at most $\frac{N}{2d}$, provided N is sufficiently large with respect to d and α and we take $\varepsilon = T/\log N$ to be small with respect to $1/\log(\alpha)$.

The first probability $1 - T\exp(-C_2\alpha N)$ is exponentially small in N, while the second probability $1 - \alpha^{-\Omega(T)}$ is exponentially small in $T = \varepsilon \log N$. Therefore, the probability of both events occurring simultaneously can be lower bounded by $1 - \alpha^{-\Omega(T)}$, by taking the constant hidden in the Ω to be slightly smaller than what is found in the previous paragraph. $\square$

Lemma 3.5.17 (Awesome implies b-good)**.** *Let $d > 0$ (as in Definition 3.5.14) be a sufficiently large constant. With probability $1 - \exp(-\Omega(B))$ over the symbol inheritance process, every awesome coupled node in $\mathcal{P}$ is b-good for at least 99% fraction of blocks $b \in [B]$.*

The figure "99%" is an arbitrary choice for simplification. It can be replaced by anything arbitrarily close to 1, which changes the constant factor of $\Omega(B)$ found in the lemma above. To prove Lemma 3.5.17, first we need a structural claim about awesome nodes:

Claim 3.5.18. *For any awesome coupled node, the subpedigree formed by it and its awesome descendants contains an induced d-ary tree that goes down to the extant level.*

Proof of Claim 3.5.18. First, we show that this subpedigree has no undirected cycles within it, which establishes the tree structure. Then, we argue that each node has d children within this subpedigree.

Suppose that there an undirected cycle within this subpedigree. We show that this implies the presence of a collision within the subpedigree, contradicting the awesomeness of all nodes in the subpedigree. Note that there must be a node within this subpedigree with a cycle in its ancestral pedigree - for instance, take the node at the lowest level within the cycle. Applying Lemma 3.4.7 to this awesome node, we see it has a collision among its ancestors, which contradicts condition 2) of Definition 3.5.14.

Now we establish that each node has at least d children in the subpedigree. An awesome coupled node v has at least d children that are d-rich, since it is d-rich itself. Furthermore, none of these children have descendants with collisions in their ancestral pedigree, so they are all awesome, which finishes the proof. $\square$

Proof of Lemma 3.5.17. Every awesome coupled node in $\mathcal{P}$ has exactly 2 distinct symbols in each block. Indeed, assume for contradiction that there is an awesome coupled node v with a block in which it only has one distinct symbol. Due to the infinite alphabet assumption, we know that we can trace any symbol in a block back to a unique founder. Hence, there must be a collision in the ancestral pedigree of v, which is a contradiction with condition 2) of (Definition 3.5.14).

Now we can proceed with showing that every awesome coupled node is b-good for 99% fraction of blocks $b \in [B]$. Fix an awesome node v and a block $b \in [B]$.

We use condition (1) of awesomeness to show that, with probability tending to 1 as $d \to \infty$, there exist two sets of three extant nodes that both have v as a joint LCA, where the first set has a symbol σ_1 in block b, and the second set has a symbol $\sigma_2 \neq \sigma_1$.

Towards this end, let us follow the inheritance of σ_1 among an induced d-ary tree of awesome descendants, as guaranteed by Claim 3.5.18. The inheritance follows a broadcast process with copy probability 1/2 on this d-ary tree. The probability that the symbol makes it to at least three distinct children of v, and this symbol in turn survives to the extant nodes can be expressed as

$$\left(1 - (1/2)^d \left(1 + d + \binom{d}{2}\right)\right) \cdot c_{d,1/2} \tag{3.8}$$

where $c_{d,1/2}$ refers to the survival probability of percolation on the d-ary tree with copy probability 1/2. The first term refers to the probability that the symbol is inherited by at least 3 of the d awesome children of v. Additionally, these three extant nodes have v as an LCA, as they have paths of inheritance from v that do not all intersect at any other node.

Naturally, Eq. (3.8) also gives the probability that σ_2 is similarly inherited. Furthermore, from standard results about Galton-Watson processes (see e.g. [94]), we know that as $d \to \infty$, $c_{d,1/2} \to 1$. Hence, we conclude that Eq. (3.8) tends to 1 as $d \to \infty$. Thus it follows from the union bound the probability that there exist two sets of three extant nodes that both have v as a lowest common ancestor, the first set has σ_1 in block b, and the second set has σ_2, also tends to 1 as $d \to \infty$.

Hence, given a specific block b, the probability that an awesome coupled node is b-good is at least 0.995. The high probability of this occurring for all blocks follows from a standard Chernoff–Hoeffding bound. □

3.6 Reconstructing the Pedigree

On the following page, we provide pseudocode for Rec-Gen which is the proposed reconstruction procedure, with details of the inner procedures following it (Collect-Symbols, Test-Siblinghood, and Assign-Parents). Note that for the first iteration of Rec-Gen, we do not need to collect symbols as the extant genetic data is given to us. Thus we simply test siblinghood at iteration $k = 1$ by using the true gene sequences.

The goal of the rest of this section is to prove the correctness of Rec-Gen. We now formally state our guarantee:

Theorem 3.6.1 (Main theorem, formal)**.** *Let $\hat{\mathcal{P}}$ be the depth-T coupled pedigree output by the algorithm* Rec-Gen*, applied to the gene sequences in $V_0(\mathcal{P})$. With probability tending towards 1 as $N \to \infty$, $\hat{\mathcal{P}}$ is an induced subpedigree of $\mathcal{P}$ such that $|V_i(\hat{\mathcal{P}})| \geq \eta(\alpha)|V_i(\mathcal{P})|$ for all levels $i \in \{0, \dots, T\}$, where $\eta(\alpha) \to 1$ as $\alpha \to \infty$. The probability is over the randomness of the coupled pedigree $\mathcal{P}$ and the inheritance procedure with parameters set as in Section 3.4.1.*

We define $\eta(\alpha) := 1 - (1/d(\alpha))$ where, for a given value of α, $d(\alpha)$ is defined to be the

Algorithm 15 Reconstruct a depth-T coupled pedigree, given extant individuals V_0.

```
1: procedure Rec-Gen($T, V_0$)
2:     $\hat{\mathcal{P}} \leftarrow (V = V_0, E = \varnothing)$                ▷ Extant Pedigree with no edges
3:     for $k = 1$ to $T$ do
4:         if $k > 1$ then
5:             for all vertices $v$ in level $k-1$ of $\hat{\mathcal{P}}$ do
6:                 Collect-Symbols($v, \hat{\mathcal{P}}$)
7:         $\hat{G} \leftarrow$ Test-Siblinghood($\hat{\mathcal{P}}$)
8:         Assign-Parents($\hat{\mathcal{P}}, \hat{G}$)
9:     return $\hat{\mathcal{P}}$
```

Algorithm 16 Empirically reconstruct the symbols of top-level node v in $\mathcal{P}$.

```
1: procedure Collect-Symbols($v, \hat{\mathcal{P}}$)
2:     for all blocks $b \in [B]$ do
3:         repeat
4:             Find extant triple $(x, y, z)$ such that:
                 1) $v$ is a joint LCA of $x, y, z$,
                 2) $x$, $y$, and $z$ all have the same symbol $\sigma$ in $b$, and
                 3) $\sigma$ is not yet recorded for block $b$ in $v$.
5:             Record the symbol $\sigma$ for block $b$ in $v$.
6:         until two distinct symbols are recorded for block $b$, or no such triple exists.
```

Algorithm 17 Perform statistical tests to detect siblinghood

```
1: procedure Test-Siblinghood(depth $(k-1)$ pedigree $\hat{\mathcal{P}}$)
2:     $V \leftarrow \{v \in V_{k-1}(\hat{\mathcal{P}}) : (\#\text{ fully recovered blocks of } v) \geq 0.99|B|\}$
3:     $E \leftarrow \varnothing$
4:     for all distinct triples $\{u, v, w\} \subset 2^V$ at level $k-1$ do
5:         if $\geq 0.21|B|$ blocks $b$ such that $\hat{s}_u(b) \cap \hat{s}_v(b) \cap \hat{s}_w(b) \neq \varnothing$ then
6:             $E \leftarrow E \cup \{u, v, w\}$
7:     return $\hat{G} = (V, E)$                ▷ 3-wise sibling hypergraph
```

Algorithm 18 Construct ancestors, given top-level 3-way sibling relationship.

```
1: procedure Assign-Parents($\mathcal{P}, G$)
2:     repeat
3:         $\mathcal{C} \leftarrow$ Any-Maximal-Clique($G$)
4:         Remove one copy of all hyper-edges in $\mathcal{C}$ from $G$.
5:         If $|\mathcal{C}| \geq d$, attach a level-$k$ parent in $\mathcal{P}$ for all nodes from $\mathcal{C}$.
6:     until no maximal cliques of size $\geq d$ remain in $G$.
```

largest value of d such that Proposition 3.5.16 holds. Observe that $d(\alpha) \to \infty$ as $\alpha \to \infty$ because Proposition 3.5.16 holds for arbitrarily large values d. Therefore, $\eta(\alpha) \to 1$ as $\alpha \to \infty$.

We make use of the following high-probability events, provided α is a large enough constant so that $d = d(\alpha)$ satisfies the hypothesis of Lemma 3.5.17, N is sufficiently large with respect to α, the total number of generations is $T = \varepsilon \log N$, where $\varepsilon = O(1/\log \alpha)$, and the gene sequence length is $B = \Omega(\log N)$.

Proposition 3.6.2 (Key Reductions)**.** *With probability tending towards* 1 *as* $N \to \infty$, *the pedigree* $\mathcal{P}$ *satisfies:*

1. *For each level* k, *each clique of* G_k *has a single parent (Lemma 3.4.13).*
2. *For each level* k, *the maximal cliques of* G_k *are edge-disjoint, in such a way that each* $v \in V_k(\mathcal{P})$ *is contained in at most two maximal cliques (Lemma 3.4.14).*
3. *Each triple* u, v, w *of nodes, has at most 3 collisions (Corollary 3.4.9), implying*
   (a) *their joint LCA is unique (Lemma 3.4.16), and*
   (b) *all inheritance paths for some node* $x \in \{u, v, w\}$ *go through the unique LCA (Lemma 3.4.17).*
4. *The fraction of overlap is at least* 24.9% *for siblings in* $\mathcal{P}$ *while for non-mutual siblings it is at most* 18.85% *(Lemmas 3.5.3 and 3.5.4).*
5. *For each level* k, *at least* $\eta(\alpha)$ *fraction of nodes in* $V_k(\mathcal{P})$ *are awesome (Proposition 3.5.16).*
6. *If* $u \in V(\mathcal{P})$ *is awesome, then it is* b*-good for* 99% *of blocks* $b \in [B]$ *(Lemma 3.5.17).*
7. *Any two* individuals *in the pedigree who are siblings overlap in at least* 49% *of their blocks (Chernoff + union bound).*

The "probability tending towards 1" portion of Theorem 3.6.1 can be quantified via a union bound on the probability of failure of any of the events in Proposition 3.6.2, while the "$|V(\hat{\mathcal{P}})| \geq \eta(\alpha)|V(\mathcal{P})|$" guarantee comes from the fact that we recover 100% of the awesome nodes in conjunction with Condition 5. With this as a simplification, we proceed with the proof of Theorem 3.6.1.

The upcoming lemma (Lemma 3.6.3) proves the correctness of the very first iteration (depth 1 from depth 0), and therefore serves as the base case. The inductive step (Lemma 3.6.4) is presented immediately afterwards. For the remainder of this section, we write $\hat{\mathcal{P}}_k$ to denote the depth-k reconstructed pedigree after the kth iteration of Rec-Gen, ($\hat{\mathcal{P}}_0$ is the depth-0 pedigree of all the extant nodes). In contrast, let $\mathcal{P}_k$ denote the subpedigree of $\mathcal{P}$ (the ground truth) induced by graded levels V_0 up to V_k.

Lemma 3.6.3. *Let $\hat{G}_0$ denote the estimated 3-regular siblinghood hypergraph for the extant nodes (line 7 of* Test-Siblinghood*). Consider the pedigree $\hat{\mathcal{P}}_1$ created by* Assign-Parents *applied to $(\hat{\mathcal{P}}_0, \hat{G}_0)$. Then there exists an injective homomorphism $\phi : \hat{\mathcal{P}}_1 \to \mathcal{P}_1$ so that the induced subgraph on $\phi(\hat{\mathcal{P}}_1)$ is isomorphic to $\hat{\mathcal{P}}_1$. Moreover, $\phi(\hat{\mathcal{P}}_1)$ contains $A_{\leq 1}$, where $A_{\leq 1}$ is the set of awesome nodes at levels ≤ 1 in $\mathcal{P}$.*

Proof. Let G_0 denote the true siblinghood hypergraph on extant nodes with at least two siblings. By Condition 4, we have that $\hat{G}_0 \cong G_0$. Since both graphs have the same set of vertices, we simply write $\hat{G}_0 = G_0$.

This gives a natural, explicit characterization of ϕ. For an extant node $v \in V_0(\mathcal{P}_1)$, define $\phi(v) = v$ so that it is the identity map on the extant. Given couple $\hat{u} \in V_1(\hat{\mathcal{P}}_1)$, define $\phi(\hat{u})$ to be the parent couple $u \in V_1(\mathcal{P}_1)$ of the children of $\hat{u}$. The condition $\hat{G}_0 \cong G_0$ implies that at least one such choice for u exists, and moreover by Condition 1, u is the unique parent.

ϕ is injective: Let $\hat{u}, \hat{v} \in V_1(\hat{\mathcal{P}}_1)$ with $\hat{u} \neq \hat{v}$. At the extant level, the maximal cliques in G_0 are vertex disjoint by Condition 2. Hence, the children of $\hat{u}$ and the children of $\hat{v}$ have empty intersection. Moreover in $\mathcal{P}_1$, vertex-disjoint maximal cliques have distinct parents. Therefore, $\phi(\hat{u}) \neq \phi(\hat{v})$, as desired.

ϕ respects edges: We already know that $(\hat{u}, v) \in E(\hat{\mathcal{P}}_1) \implies (\phi(\hat{u}), v) \in E(\mathcal{P}_1)$. Now suppose that $(\phi(\hat{u}), v)$ is an edge in $\mathcal{P}_1$ for $\hat{u} \in V_1(\hat{\mathcal{P}}_1)$ and $v \in V_0(\hat{\mathcal{P}}_1)$. Since u is in the image of ϕ, it follows that u has at least 3 children w, x, y that passed the siblings test in our algorithm. If v is one of w, x, y, we're done, so suppose not. By Condition 3.5.3, the extant triples $\{v, w, x\}$, $\{v, x, y\}$, and $\{v, w, y\}$ all have at least 24% overlap. Therefore, v, w, x, y form a clique in $\hat{G}_0$, and line 5 of Assign-Parents states that $\hat{u}$ is a parent of all four, so $(\hat{u}, v)$ is an edge in $\hat{\mathcal{P}}_1$.

The image of ϕ contains the awesome nodes in $\mathcal{P}_1$: This part is trivially true for the extant nodes, so consider only the awesome nodes $A_1 \subset V_1(\mathcal{P}_1)$. By definition, any awesome node $u \in V_1(\mathcal{P}_1)$ is d-rich. Since $d \geq 3$, the children of u form a maximal clique of size at least 3 in G_0. Therefore, Assign-Parents creates a parent $\hat{u}$ for these children in $\hat{\mathcal{P}}_1$, which gives the pre-image of u. □

Lemma 3.6.4. *Let $k \geq 2$ and suppose that we are given $\hat{\mathcal{P}}_{k-1}$. Assume that there exists an injective homomorphism $\phi : \hat{\mathcal{P}}_{k-1} \to \mathcal{P}_{k-1}$ which satisfies*

1. *$\phi|_{\hat{\mathcal{P}}_0} \equiv Id$,*
2. *$\phi(\hat{\mathcal{P}}_{k-1}) \subset \mathcal{P}_{k-1}$ induces a subgraph isomorphic to $\hat{\mathcal{P}}_{k-1}$, and*
3. *$\phi(\hat{\mathcal{P}}_{k-1})$ contains the awesome nodes in sets $A_0, A_1, \ldots, A_{k-1}$.*

Let $\hat{\mathcal{P}}_k$ be the level-k extension of $\hat{\mathcal{P}}_{k-1}$, via lines 4 through 7 of Rec-Gen. *Then there exists a level-k extension of the map $\phi : \hat{\mathcal{P}}_k \to \mathcal{P}_k$ with the same properties.*

We prove this in two stages. The first part (Lemma 3.6.5) asserts that we reconstruct the sibling relationships correctly, while the latter (Lemma 3.6.11) assures that the cliques of this estimated siblinghood hypergraph are actually the faithful, "largest possible" groupings of siblings.

Lemma 3.6.5. *Assume the hypotheses of Lemma 3.6.4, and let $\hat{G}_{k-1}$ be the estimated siblinghood hypergraph constructed by* Test-Siblinghood, *line 7, on input $\hat{\mathcal{P}}_{k-1}$. Then the subgraph of G_{k-1}*

induced by $\phi(\hat{G}_{k-1})$ is isomorphic to $\hat{G}_{k-1}$, and moreover $\phi(\hat{G}_{k-1})$ contains all of the awesome nodes A_{k-1} at level $k-1$.

The upcoming statements (Claim 3.6.7, Claim 3.6.8 and Claim 3.6.9) are pivotal for the proof of Lemma 3.6.5.

Definition 3.6.6. *For an awesome node $u \in \mathcal{P}_k$, its* awesome subtree *is the subgraph of $\mathcal{P}_k$ that is the union of all paths from u to extant nodes that consist entirely of awesome nodes.*

Claim 3.6.7. *Suppose that there is a reconstruction map $\phi : \hat{\mathcal{P}}_{k-1} \to \mathcal{P}_{k-1}$ satisfying the hypotheses in Lemma 3.6.4. Then for any awesome node $u = \phi(\hat{u}) \in V_{k-1}(\mathcal{P}_{k-1})$, its awesome subtree S_u satisfies $\phi^{-1}(S_u) = \mathsf{desc}(\hat{u})$.*

Proof of Claim 3.6.7. Note that Line 5 of Assign-Parents ensures that every node in $\hat{\mathcal{P}}_{k-1}$ is d-rich. Since ϕ is an injective homomorphism, it follows that every node in $\phi(\mathsf{desc}(\hat{u}))$ is also d-rich in $\mathcal{P}$. Furthermore, u being awesome implies that all of its descendants are awesome in $\mathcal{P}$, since none of its descendants can have collisions in its ancestral pedigree (Definition 3.5.14). By the definition of the awesome subtree (Definition 3.6.6), it holds that $\phi(\mathsf{desc}(\hat{u})) \subseteq S_u$.

For the other direction ($\phi(\mathsf{desc}(\hat{u})) \supseteq S_u$), let $v \in V_0(\mathcal{P})$ be an extant node so that there is a path from u to v consisting only of awesome nodes. By condition 3 of Lemma 3.6.4, all of the nodes along this path are in the image of ϕ. □

Claim 3.6.8. *Let ϕ be as in Lemma 3.6.4, and let $u = \phi(\hat{u})$ for some $\hat{u} \in V_{k-1}(\hat{\mathcal{P}}_{k-1})$. Suppose that in block b the symbols $\hat{\sigma}_1$ and $\hat{\sigma}_2$ are recovered for $\hat{u}$ by applying Algorithm 1 to $\hat{u}$. Then it holds that u also has symbols $\hat{\sigma}_1, \hat{\sigma}_2$ in block b.*

Proof of Claim 3.6.8. For $i = 1, 2$, suppose that nodes $x_i, y_i, z_i \in V_0(\hat{\mathcal{P}}_0) = V_0(\mathcal{P})$ have the symbol $\hat{\sigma}_i$ in block b and are used by Collect-Symbols to recover $\hat{\sigma}_i$ in block b of $\hat{u}$. Recall that x_i, y_i, z_i are all descended from distinct children of $\hat{u}$. Let $\phi(\hat{\mathcal{P}}_{k-1})$ induce subpedigree $\mathcal{Q}$ in $\mathcal{P}$.

By the hypotheses of Lemma 3.6.4, $\mathcal{Q} \cong \hat{\mathcal{P}}_{k-1}$ and so u must be a common ancestor of x_i, y_i, z_i in $\mathcal{Q}$. By line 4 of Collect-Symbols and because $\mathcal{Q} \cong \hat{\mathcal{P}}_{k-1}$, $\hat{u}$ – and therefore u

– is their joint LCA. With respect to $\mathcal{P}$, Conditions 3a and 3b tell us the much stronger condition that u is their only LCA, and that all paths in $\mathcal{P}$ from any common ancestor of x_i, y_i, z_i to x_i (without loss of generality) must pass through u. Therefore, if x_i, y_i, z_i all inherit symbols $\hat{\sigma}_i$ in block b, the symbol $\hat{\sigma}_i$ must have passed through block b of u via the infinite symbols assumption. □

Claim 3.6.9. *Let ϕ be as in Lemma 3.6.4, and let $u = \phi(\hat{u})$ for some $\hat{u} \in V_{k-1}(\hat{\mathcal{P}}_{k-1})$. Suppose that u is awesome in $\mathcal{P}$. If u is b-good and has symbols σ_1, σ_2 in block b, then* Collect-Symbols *recovers the symbols σ_1 and σ_2 for $\hat{u}$ in block b.*

Proof of Claim 3.6.9. By Claim 3.6.8, we only need to show that at least two symbols in block b are reconstructed by Collect-Symbols applied to $\hat{u}$. Note that b-goodness implies $\sigma_1 \neq \sigma_2$.

By b-goodness of u, as in the proof of Lemma 3.5.17, there is a witnessing triple for each of the σ_i contained in the extant of the awesome subtree S_u. By Claim 3.6.7, $\text{desc}(\hat{u})$ also contains these witnesses. Since extant nodes are the exact same in $\mathcal{P}$ compared to $\hat{\mathcal{P}}_{k-1}$ by hypothesis 1 of Lemma 3.6.4, Collect-Symbols applied to $\hat{u}$ recovers σ_1, σ_2 in block b. □

Claim 3.6.10. *Couples that consist of two siblings in the true pedigree do not appear in the siblinghood graph $\phi(\hat{G}_{k-1})$.*

Proof. Denote the coupled vertex as v. We note that any two sibling *individuals* must overlap in at least 49% fraction of blocks in the true pedigree (Condition 7). We know that any symbol that we retrieve for v via the symbol collection must actually come from v, since the symbols come from triples of extant nodes with v as their joint LCA (Conditions 3a and 3b). Hence, for any couples node consisting of two siblings, we can retrieve two symbols for them in at most 51% of blocks. This means that they do not appear in the reconstructed siblinghood graph (Line 2), and so they do not appear in $\hat{G}_{k-1}$. Due to the fact that $\phi(\hat{G}_{k-1}))$ induces a subgraph isomorphic to $\hat{G}_{k-1}$, they do not appear in $\phi(\hat{G}_{k-1})$ either. □

Since these nodes do not appear in our reconstructed siblings graph, we do not reconstruct parents for these nodes. We note that this is fine, as such nodes are not awesome couples nodes; hence, their ancestors are also not awesome, and it is fine not to reconstruct them. With this, we begin the proof of Lemma 3.6.5.

Proof of Lemma 3.6.5. For this claim, we use the convention that the true siblinghood graph G_{k-1} does not have any hyperedges that contain a single couples node twice; this fact is true of the reconstructed siblinghood graph $\hat{G}_{k-1}$ by construction (Claim 3.6.10). It is hence crucial in proving the isomorphism, and does not harm our end guarantee, since we do not miss out on reconstructing any awesome node with this omission by the discussion above. So, graph isomorphism boils down to showing that any triple of distinct nodes $\{u, v, w\}$ has a hyper edge on it iff they are siblings in the true siblinghood graph, since all hyperedges have multiplicity 1.

By assumption, $\phi : \hat{G}_{k-1} \to G_{k-1}$ is injective. To first see that ϕ is a hypergraph homomorphism, let $\hat{u}, \hat{v}, \hat{w} \in V_{k-1}(\hat{\mathcal{P}}_{k-1})$ be distinct nodes satisfying line 2 of Test-Siblinghood, and let $u = \phi(\hat{u}), v = \phi(\hat{v})$, and $w = \phi(\hat{w})$ denote their counterparts in $\mathcal{P}$.

Suppose that u, v, w are not mutually siblings. By Condition 4, u, v, w have at most $0.1885|B|$ mutually overlapping blocks. By Claim 3.6.8, for all $\hat{x} \in \{\hat{u}, \hat{v}, \hat{w}\}$, the symbols reconstructed for $\hat{x}$ in block b using Collect-Symbols are a subset of the symbols in block b of $x := \phi(\hat{x}) \in \{u, v, w\}$. Therefore, $\hat{u}, \hat{v}, \hat{w}$ have mutually overlapping symbols in at most $0.1885|B|$ blocks. Since $0.1885 < 0.21$, Test-Siblinghood does not place a hyperedge between $\hat{u}, \hat{v}, \hat{w}$ in $\hat{G}_1$.

To show that the induced subgraph $\phi(\hat{G}_{k-1})$ is isomorphic to $\hat{G}_{k-1}$, it remains to show that if u, v, w are distinct nodes that are mutual siblings in $\mathcal{P}$, then $\{\hat{u}, \hat{v}, \hat{w}\}$ is a hyperedge in $\hat{G}_{k-1}$. This suffices because no couples node appears twice in the true siblings graph, by construction. Note that 99% of the blocks of $\hat{u}, \hat{v}, \hat{w}$ were recovered by Collect-Symbols by the definition of $\hat{G}_{k-1}$, and by Claim 3.6.8, the symbols of $\hat{u}, \hat{v}, \hat{w}$ in block b are a subset of the symbols of u, v, w, respectively, in block b. By Condition 4, the mutual overlap between the siblings u, v, w is at least $0.249|B|$. Thus, by a union bound on the occurrence of 1%-fraction of unrecovered blocks, the mutual overlap between $\hat{u}, \hat{v}, \hat{w}$ is at least $(0.249 - 0.03)|B| \geq 0.21|B|$. Therefore, Test-Siblinghood constructs a hyperedge on $\hat{u}, \hat{v}, \hat{w}$, as desired.

Finally, we show that the awesome nodes A_{k-1} are fully contained in $\phi(\hat{G}_{k-1})$. By Condition 6, awesome nodes are b-good. Now apply Claim 3.6.9, to conclude that Collect-Symbols reconstructs 99% of the blocks in each awesome node u, so $u \in \hat{G}_{k-1}$ according to Line 2 of Test-Siblinghood. □

Lemma 3.6.11. *Let $\mathscr{C}$ denote the maximal (hyper)cliques in the subgraph of G_{k-1} induced by $\phi(\hat{G}_{k-1})$, and let $\mathscr{C}_{algo}$ denote the (hyper)cliques probed by* Assign-Parents *applied to $\hat{G}_{k-1}$. Given $\mathcal{C} \in \mathscr{C}_{algo}$, define $\phi(\mathcal{C})$ to be the set given by the image of $\mathcal{C}$ under ϕ. Then ϕ is a bijection between $\mathscr{C}_{algo}$ and $\mathscr{C}$.*

Proof. By Lemma 3.6.5, the subgraph H induced by $\phi(\hat{G}_{k-1})$ is isomorphic to $\hat{G}_{k-1}$. Hence, it suffices to show that the cliques probed by Assign-Parents applied to H are precisely the maximal cliques of H. Recall that by Condition 2, the maximal cliques in H are edge-disjoint, and every node of H is involved in at most 2 cliques.

It is helpful to imagine the cliques $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_M \in \mathscr{C}_{\text{algo}}$ as being listed out in the same order that they are probed by Assign-Parents, indexed by timesteps $m = 1, 2, \ldots, M$. Let $H^{(0)} = H$, and let $H^{(m)}$ denote the result of removing the edges of the clique $\mathcal{C}_t$ from $H^{(m-1)}$.

We argue that for all m, the graph $H^{(m)}$ is a union of edge-disjoint maximal cliques, and any two maximal cliques intersect in at most a single vertex. The base case $m = 1$ is true by Condition 2. This holds for $m > 1$ because the above property is preserved when all of the edges are removed from a single maximal clique in $H^{(m-1)}$. Moreover, for all m, the maximal cliques in $H^{(m)}$ are the same as those of $H^{(m-1)}$ but with a single maximal clique $\mathcal{C}_m$ in $H^{(m-1)}$ removed. Hence, it also follows by induction that for all m, the maximal clique $\mathcal{C}_m$ in $H^{(m-1)}$ is also a maximal clique in H.

Since Assign-Parents terminates at the first time M when $H^{(M)}$ has no hyperedges, we conclude that $\mathcal{C}_1, \ldots, \mathcal{C}_M$ are *all* of the maximal cliques in H, as desired. □

Proof of Lemma 3.6.4. We first extend the definition of ϕ to level k. For $\hat{u} \in V_k(\hat{\mathcal{P}}_k)$, we define $\phi(\hat{u}) \in V_k(\mathcal{P}_k)$ as follows. Let $\hat{\mathcal{C}} \subset V_{k-1}(\hat{\mathcal{P}}_k)$ denote the children of $\hat{u}$. By Lemmas 3.6.5 and 3.6.11, $\phi(\hat{\mathcal{C}})$ is a clique in G_{k-1}. Define $\phi(\hat{u}) \in V_k(\mathcal{P}_k)$ to be the parent of the children of the clique $\phi(\hat{\mathcal{C}})$ in $\mathcal{P}$. The map ϕ is well-defined at level k because of Condition 1. It remains to show that ϕ is an isomorphism onto its image, and moreover that its image contains all of the awesome nodes at level k.

The map ϕ is injective: We know this is true for $\phi\big|_{\hat{\mathcal{P}}_{k-1}}$, so it suffices to consider injectivity of ϕ when restricted to the nodes at level k in $\hat{\mathcal{P}}_k$. Let $\hat{u}, \hat{v} \in V_k(\hat{\mathcal{P}}_k)$ with $\hat{u} \neq \hat{v}$. Let $\mathcal{C}$ (resp., $\mathcal{C}'$) denote the maximal clique in $\hat{G}_{k-1}$ that consists of the children of $\hat{u}$ (resp., $\hat{v}$). By Lemma 3.6.11, $\phi(\mathcal{C})$ and $\phi(\mathcal{C}')$ are distinct maximal cliques in the induced subgraph $\phi(\hat{G}_{k-1})$, and therefore, are contained in distinct maximal cliques in G_{k-1}. Distinct maximal cliques in G_{k-1} have distinct parents, so by the definition of ϕ, we conclude that $\phi(\hat{u}) \neq \phi(\hat{v})$, as desired.

The map ϕ is edge-preserving: Suppose that $(\hat{u}, \hat{v})$ is an edge in $\hat{\mathcal{P}}_k$ with $\hat{u} \in V_k(\hat{\mathcal{P}}_k)$ and $\hat{v} \in V_{k-1}(\hat{\mathcal{P}}_k)$. Consider the maximal clique $\hat{\mathcal{C}}$ containing $\hat{v}$ in $\hat{G}_{k-1}$. By Lemma 3.6.11, $\phi(\hat{\mathcal{C}})$ is a maximal clique in the induced subgraph $\phi(\hat{G}_{k-1}) \subset G_{k-1}$, and by construction of ϕ, the parent of $\phi(\hat{\mathcal{C}})$ is $\phi(\hat{u})$. Therefore, the edge $(\phi(\hat{u}), \phi(\hat{v}))$ is in the pedigree $\mathcal{P}_k$.

Suppose now that the edge $(u, v) = (\phi(\hat{u}), \phi(\hat{v}))$ is in the pedigree $\mathcal{P}_k$. Consider the maximal clique $\mathcal{C}' \subset G_{k-1}$ containing v. By Lemma 3.6.11, $\mathcal{C} := \phi^{-1}(\mathcal{C}') = \{x \in \hat{\mathcal{P}}_k : \phi(x) \in \mathcal{C}'\}$ is a maximal clique in $\hat{G}_{k-1}$. By Lemma 3.6.11 and the construction in Assign-Parents, we conclude that the parent of $\hat{v}$ in $\hat{\mathcal{P}}_k$ is mapped to u under ϕ. By injectivity of ϕ, this parent is precisely $\phi^{-1}(u) = \hat{u}$. Therefore, $(\hat{u}, \hat{v})$ is an edge in $\hat{\mathcal{P}}_k$.

The image of ϕ contains the awesome nodes in $\mathcal{P}_k$: It suffices to prove the statement for the awesome nodes at level k, which we denote by A_k. Suppose that u is an awesome node at level k of $\mathcal{P}$. By awesomeness, u has at least d awesome children. Let $\mathcal{C}'$ denote the clique in G_{k-1} given by the awesome children of u. By Lemmas 3.6.5 and 3.6.11, $\mathcal{C} := \phi^{-1}(\mathcal{C}')$ satisfies $|\mathcal{C}| = |\mathcal{C}'| \geq d$ because all of the awesome children up to level $k-1$ are in the image of ϕ, by the inductive hypotheses. By Lemma 3.6.11, the maximal clique $\tilde{\mathcal{C}}$ containing $\mathcal{C}$ in $\hat{G}_{k-1}$ satisfies that $\phi(\tilde{\mathcal{C}})$ are all children of u. By the definition of Assign-Parents and ϕ at level k, we conclude that a parent $\hat{u}$ is constructed for $\tilde{\mathcal{C}} \supset \mathcal{C}$ and $\phi(\hat{u}) = u$, as desired. $\square$

Acknowledgments We thank Vishesh Jain for many helpful discussions.

Chapter 4

Corruption Detection

4.1 Introduction

4.1.1 Corruption Detection and Problem Set-up

Starting from this chapter, we formulate inference problems inspired by social science. Previously, our inference questions have only involved a passive observer, who sees data generated by individuals but does not influence the reporting of data in any way. However, when we look at problems of inference in the setting of social science, it is natural to broaden the motivations of the central parties and allow them to engage in the underlying process. For the remainder of this thesis, we look at inference problems in social science where one or more agents wish to influence the underlying process to promote their own agenda.

In this chapter we study the problem of identifying truthful nodes in networks, in the model of *corruption detection on networks* posed by Alon, Mossel, and Pemantle [6]. In this model, we have a network represented by a (possibly directed) graph. Nodes can be *truthful* or *corrupt*. Each node audits its outgoing neighbors to see whether they are truthful or corrupt, and sends reports of their identities to a central agency. The central agent, who is not part of the graph, aggregates the reports and uses them to identify truthful and corrupt nodes. Truthful nodes report truthfully (and correctly) on their neighbors, while corrupt nodes have no such restriction: they can assign arbitrary reports to their neighbors, regardless of whether their neighbors are truthful or corrupt, and coordinate their efforts with each other to prevent the central agency from gathering useful information.

In [6], the authors consider the problem of recovering the identities of almost all nodes in a network in the presence of many corrupt nodes; specifically, when the fraction of corrupt nodes can be very close to 1/2. They call this the *corruption detection* problem. They show that the central agency can recover the identity of most nodes correctly even in certain bounded-degree graphs, as long as the underlying graph is a sufficiently good expander. The required expansion properties are known to hold for a random graph or Ramanujan graph of sufficiently large (but constant) degree, which yields undirected graphs that are amenable to corruption detection. Furthermore, they show that some level of expansion is necessary for identifying truthful nodes, by demonstrating that the corrupt nodes can stop the central agency from identifying any truthful node when the graph is a very bad expander (e.g. a cycle), even if the corrupt nodes only make up 0.01 fraction of the network.

This establishes that very good expanders are very good for corruption detection, and very bad expanders can be very bad for corruption detection. We note that this begs the question of how effective graphs that do not fall in either of these categories are for corruption detection. In the setting of [6], we could ask the following: given an *arbitrary* undirected graph, what is the smallest number of corrupt nodes that can prevent the identification of almost all nodes? When there are fewer than this number, can the central agency *efficiently* identify almost all nodes correctly? Alon, Mossel, and Pemantle study these questions for the special cases of highly expanding graphs and poorly expanding graphs, but do not address general graphs.

Additionally, [6] considers corruption detection when the corrupt agencies can choose their locations and collude arbitrarily, with no bound on their computational complexity. This is perhaps overly pessimistic: after all, it is highly unlikely that corrupt agencies can solve NP-hard problems efficiently and if they can, thwarting their covert operations is unlikely to stop their world domination. We suggest a model that takes into account computational considerations, by factoring in the computation time required to select the nodes in a graph that a corrupt party chooses to control. This yields the following question from the viewpoint of a corrupt party: given a graph, can a corrupt party compute the smallest set of nodes it needs to corrupt *in polynomial time*?

In addition to being natural from a mathematical standpoint, these questions are also well-motivated socially. It would be naïve to assert that we can weed out corruption in the real world by simply designing auditing networks that are expanders. Rather, these networks may already be formed, and infeasible to change in a drastic way. Given this, we are less concerned with finding certain graphs that are good for corruption detection, but rather discerning how good *existing* graphs are; specifically, how many corrupt nodes they can tolerate. In particular, since the network structure could be out of the control of the central agency, algorithms for the central agency to detect corruption on arbitrary graphs seem particularly important.

It is also useful for the *corrupt* agency to have an algorithm with guarantees for any graph. Consider the following example of a corruption detection problem from the viewpoint of a corrupt organization. Country A wants to influence policy in country B, and wants to figure out the most efficient way to place corrupted nodes within country B to make this happen. However, if the central government of B can confidently identify truthful nodes, they can weight those nodes' opinions more highly, and thwart country A's plans. Hence, the question country A wants to solve is the following: given the graph of country B, can country A compute the optimal placement of corrupt nodes to prevent country B from finding truthful nodes? We note that in this question, too, the graph of country B is fixed, and hence, country A would like to have an algorithm that takes as input *any* graph and computes the optimal way to place corrupt nodes in order to hide all the truthful nodes.

We study the questions above for a variant of the corruption detection problem in [6], in which the goal of the central agency is to find a single truthful node. While this goal is less ambitious than the goal of identifying almost all the nodes, we think it is a very natural question in the context of corruption. For one, if the central agency can find a single truthful node, they can use the trusted reports from that node to identify more truthful and corrupt nodes that it might be connected to. The central agency may additionally weight the opinions of the truthful nodes more when making policy decisions (as alluded to in the example above), and can also incentivize truthfulness by rewarding truthful nodes that it finds and giving them more influence in future networks if possible (by increasing their out-degrees). Moreover, our proofs and results extend to finding larger number of truthful nodes as we discuss below.

Our results stem from a tie between the problem of finding a single truthful node in a graph and a measure of vertex separability of the graph. This tie not only yields an efficient and relatively effective algorithm for the central agency to find a truthful node, but also allows us to relate corrupt party's strategy to the problem of finding a good vertex separator for the graph. Hence, by analyzing the purely graph-theoretic problem of finding a good vertex separator, we can characterize the difficulty of finding a good set of nodes to corrupt. Similar notions of vertex separability have been studied previously (e.g. [105, 128, 17]), and we prove NP-hardness for the notion relevant to us assuming the Small Set Expansion Hypothesis (SSEH). The *Small Set Expansion Hypothesis* is a hypothesis posed by Raghavendra and Steurer [137] that is closely related to the famous Unique Games Conjecture of Khot [90]. In fact, [137] shows that the SSEH implies the Unique Games Conjecture. The SSEH yields hardness results that are not known to follow directly from the UGC, especially for graph problems like sparsest cut and treewidth ([138] and [9] respectively), among others.

4.1.2 Our Results

We now outline our results more formally. We analyze the variant of corruption detection where the central agency's goal is to find a single truthful node. First, we study how effectively the central agency can identify a truthful node on an arbitrary graph, given a set of reports. Given an undirected graph[1] G, we let $m(G)$ denote the minimal number of corrupted nodes required to stop the central agency from finding a truthful node, where the minimum is taken over all strategies of the corrupt party (not just computationally bounded ones). We informally call $m(G)$ the "critical" number of corrupt nodes for a graph G. Then, we show the following:

Theorem 4.3.2. *Fix a graph G and suppose that the corrupt party has a budget $b \leq m(G)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without knowledge of either $m(G)$ or b. Furthermore, the central agency's algorithm runs in linear time (in the number of edges in the graph G).*

Next, we consider the question from the viewpoint of the corrupt party: can the corrupt party efficiently compute the most economical way to allocate nodes to prevent the central agency from finding a truthful node? Concretely, we focus on a natural decision version of the question: given a graph G and a upper bound on the number of possible corrupted nodes k, can the corrupt party prevent the central agency from finding a truthful node?

We actually focus on an easier question: can the corrupt party accurately compute $m(G)$, the minimum number of nodes that they need to control to prevent the central agency from finding a truthful node? Not only do we give evidence that computing $m(G)$ exactly is computationally hard, but we also provide evidence that $m(G)$ is hard to approximate. Specifically, we show that approximating $m(G)$ to any constant factor is NP-hard under the Small Set Expansion Hypothesis (SSEH); or in other words, that it is SSE-hard.

Theorem 4.3.5. *For every $\beta > 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V, E)$, it is SSE-hard to distinguish between the case where $m(G) \leq \epsilon \cdot |V|$ and $m(G) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes for a graph to within any constant factor is SSE-hard.*

This Theorem immediately implies the following Corollary 4.3.9.

[1]Unless explicitly specified, all graphs are undirected by default.

Corollary 4.3.9. *Assume the SSE Hypothesis and that P ≠ NP. Fix any $\beta > 1$. There does not exist a polynomial-time algorithm that takes as input an arbitrary graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\beta \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

We note that in Corollary 4.3.9, the bad party's input is only the graph G: specifically, they do not have knowledge about the value of $m(G)$.

Our proof for Theorem 4.3.5 is similar to the proof of Austrin, Pitassi, and Wu [9] for the SSE-hardness of approximating treewidth. This is not a coincidence: in fact, the "soundness" in their reduction involves proving that their graph does not have a good 1/2 vertex separator, where the notion of vertex separability (from [23]) is very related to the version we use to categorize the problem of hiding a truthful vertex. We give the proof of Theorem 4.3.5 in Section 4.3.2.

However, if one allows for an approximation factor of $O(\log|V|)$, then $m(G)$ can be approximated efficiently. Furthermore, this yields an approximation algorithm that the corrupt party can use to find a placement that hinders detection of a truthful node.

Theorem 4.3.11. *There is a polynomial-time algorithm that takes as input a graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\log|V| \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

The proof of Theorem 4.3.11, given in Section 4.3.2, uses a bi-criterion approximation algorithm for the k-vertex separator problem given by [105]. As alluded to in Section 4.1.1, Theorems 4.3.5 and 4.3.11 both rely on an approximate characterization of $m(G)$ in terms of a measure of vertex separability of the graph G, which we give in Section 4.3.

Additionally, we note that we can adapt Theorems 4.3.2 and 4.3.5 (as well as Corollary 4.3.9) to a more general setting, where the central agency wants to recover some arbitrary number of truthful nodes, where the number of nodes can be proportional to the size of the graph. We describe how to modify our proofs to match this more general setting in Section 4.5.

Together, Theorems 4.3.2 and 4.3.5 uncover a surprisingly positive result for corruption detection: it is computationally easy for the central agency to find a truthful node when the number of corrupted nodes is only somewhat smaller than the "critical" number for the underlying graph, but it is in general computationally hard for the corrupt party to hide the truthful nodes even when they have a budget that far exceeds the "critical" number for the graph.

Results for Directed Graphs As noted in [6], it is unlikely that real-world auditing networks are undirected. For example, it is likely that the FBI has the authority to audit the Cambridge police department, but it is also likely that the reverse is untrue. Therefore, we would like the central agency to be able to find truthful nodes in directed graphs in addition to undirected graphs. We notice that the algorithm we give in Theorem 4.3.2 extends naturally to directed graphs.

Theorem 4.4.4. *Fix a directed graph D and suppose that the corrupt party has a budget $b \leq m(D)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without the knowledge of either $m(D)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

The proof of Theorem 4.4.4 is similar to the proof of Theorem 4.3.2, and effectively relates the problem of finding a truthful node on directed graphs to a similar notion of vertex separability, suitably generalized to directed graphs.

Results for Finding An Arbitrary Number of Good Nodes In fact, the problem of finding one good node is just a special case of finding an arbitrary number of good nodes, g, on the graph G. We define $m(G,g)$ as the minimal number of bad nodes required to prevent the identification of g good nodes on the graph G. We relate it to an analogous vertex separation notion, and prove the following two theorems, which are extensions of Theorems 4.3.2 and 4.3.5 to this setting.

Theorem 4.5.3. *Fix a graph G and the number of good nodes to recover, g. Suppose that the corrupt party has a budget $b \leq m(G,g)/2$. If $g < |V| - 2b$, then the central agency can identify g truthful nodes, regardless of the strategy of the corrupt party, and without knowledge either of $m(G,g)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

Theorem 4.5.5. *For every $\beta > 1$ and every $0 < \delta < 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V,E)$, it is SSE-hard to distinguish between the case where $m(G,\delta|V|) \leq \epsilon \cdot |V|$ and $m(G,\delta|V|) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes such that it is impossible to find $\delta|V|$ good nodes within any constant factor is SSE-hard.*

The proof of Theorem 4.5.5 is similar to the proof of Theorem 4.3.2, and the hardness of approximation proof also relies on the same graph reduction and SSE conjecture. Proofs are presented in Section 4.5.

4.1.3 Related Work

The model of corruptions posed by [6] is identical to a model first suggested by Preparata, Metze, and Chien [135], who introduced the model in the context of detecting failed components in digital systems. This work (as well as many follow-ups, e.g. [87, 99]) looked at the problem of characterizing which networks can detect a certain number of corrupted nodes. Xu and Huang [166] give necessary and sufficient conditions for identifying a single corrupted node in a graph, although their characterization is not algorithmically efficient. There are many other works on variants of this problem (e.g. [156, 37]), including recovering node identities with one-sided or two-sided error probabilities in the local reports [111] and adaptively finding truthful nodes [72].

We note that our model of a computationally bounded corrupt party and our stipulation that the graph is fixed ahead of time rather than designed by the central agency, which are our main contributions to the model, seem more naturally motivated in the setting of corruptions than in the setting of designing digital systems. Even the question of identifying a single truthful node could be viewed as more naturally motivated in the setting of corruptions than in the setting of diagnosing systems. We believe there are likely more interesting theoretical questions to be discovered by approaching the PMC model through a corruptions lens.

The identifiability of a single node in the corruptions setting was studied in a recent paper of Mukwembi and Mukwembi [119]. They give a linear time greedy algorithm to recover the identify of a single node in many graphs, *provided that corrupt nodes always report other corrupt nodes as truthful*. Furthermore, this assumption allows them to reduce identifying all nodes to identifying a single node. They argue that such an assumption is natural in the context of corruptions, where corrupt nodes are selfishly incentivized not to out each other. However, in our setting, corrupt nodes can not only betray each other, but are in fact incentivized to do so for the good of the overarching goal of the corrupt party (to prevent the central agency from identifying a truthful node). Given [119], it is not a surprise that the near-optimal strategies we describe for the corrupt party in this paper crucially rely on the fact that the nodes can report each other as corrupt.

Our problem of choosing the best subset of nodes to corrupt bears intriguing similarities to the problem of influence maximization studied by [89], where the goal is to find an optimal set of nodes to target in order to maximize the adoption of a certain technology or product. It is an interesting question to see if there are further similarities between these two areas. Additionally, social scientists have studied corruption extensively (e.g.[50], [125]), though to the best of our knowledge they have not studied it in the graph-theoretic way that we do in this paper.

4.1.4 Comparison to Corruption in Practice

Finally, we must address the elephant in the room. Despite our theoretical results, corruption *is* prevalent in many real-world networks, and yet in many scenarios it is not easy to pinpoint even a single truthful node. One reason for that is that some of assumptions do not seem to hold in some real world networks. For example, we assume that audits from the truthful nodes are not only non-malicious, but also perfectly reliable. In practice this assumption is unlikely to be true: many truthful nodes could be non-malicious but simply unable to audit their neighbors accurately. Further assumptions that may not hold in some scenarios include the notion of a central agency that is both uncorrupted and has access to reports from every agency, and possibly even the assumption that the number of corrupt nodes is less than $|V|/2$. In addition, networks G may have very low critical numbers $m(G)$ in practice. For example, there could be a triangle (named, “President”, “Congress” and “Houses”) that is all corrupt and cannot be audited by any agent outside the triangle. It is thus plausible that a corrupt party could use the structure of realistic auditing networks for their corruption strategy to overcome our worst-case hardness result.

While this points to some shortcomings of our model, it also points out ways to change policy that would potentially bring the real world closer to our idealistic scenario, where a corrupt party has a much more difficult computational task than the central agency. For example, we can speculate that perhaps information should be gathered by a transparent centralized agency, that significant resources should go into ensuring that the centralized agency is not corrupt, and that networks ought to have good auditing structure (without important agencies that can be audited by very few nodes).

4.2 Preliminaries

4.2.1 General Preliminaries

We denote undirected graphs by $G = (V, E)$, where V is the vertex set of the graph and E is the edge set. We denote directed graphs by $D = (V, E_D)$. When the underlying graph is clear, we may drop the subscripts. Given a vertex u in an undirected graph G, we let $\mathcal{N}(u)$ denote the *neighborhood* (set of neighbors) of the vertex in G. Similarly, given a vertex u in a directed graph D, let $\mathcal{N}(u)$ denote the set of *outgoing* neighbors of u: that is, vertices $v \in V$ such that $(u, v) \in E_D$.

Vertex Separator

Definition 4.2.1. ***(k-vertex separator)****([128],[17]) For any $k \geq 0$, we say a subset of vertices $U \subseteq V$ is k-vertex separator of a graph G, if after removing U and incident edges, the remaining graph forms a union of connected components, each of size at most k.*

Furthermore, let

$$S_G(k) = \min\Big(|U| : U \textit{ is a k-vertex separator of } G\Big)$$

denote the size of the minimal k-vertex separator of graph G.

Small Set Expansion Hypothesis

In this section we define the Small Set Expansion (SSE) Hypothesis introduced in [137]. Let $G = (V,E)$ be an undirected d-regular graph.

Definition 4.2.2 (Normalized edge expansion)**.** *For a set $S \subseteq V$ of vertices, denote $\Phi_G(S)$ as the normalized edge expansion of S,*

$$\Phi_G(S) = \frac{|E(S, V\backslash S)|}{d|S|},$$

where $|E(S,V\backslash S)|$ is the number of edges between S and $V\backslash S$.

The Small Set Expansion Problem with parameters η and δ, denoted SSE(η,δ), asks whether G has a small set S which does not expand or all small sets of G are highly expanding.

Definition 4.2.3 ((SSE(η,δ)))**.** *Given a regular graph $G = (V,E)$, distinguish between the following two cases:*

- ***Yes*** *There is a set of vertices $S \subseteq V$ with $S = \delta|V|$ and $\Phi_G(S) \leq \eta$*
- ***No*** *For every set of vertices $S \subseteq V$ with $S = \delta|V|$ it holds that $\Phi_G(S) \geq 1-\eta$*

The *Small Set Expansion Hypothesis* is the conjecture that deciding SSE(η,δ) is NP-hard.

Conjecture 4.2.4 (Small Set Expansion Hypothesis [137]). *For every $\eta > 0$, there is a $\delta > 0$ such that SSE(η, δ) is NP-hard.*

We say that a problem is *SSE-hard* if it is at least as hard to solve as the SSE problem. The form of conjecture most relevant to our proof is the following "stronger" form of the SSE Hypothesis. [138] showed that the SSE-problem can be reduced to a quantitatively stronger form of itself. In order to state this version, we first need to define the *Gaussian noise stability*.

Definition 4.2.5. *(Gaussian Noise Stability) Let $\rho \in [-1,1]$. Define $\Gamma_\rho : [0,1] \mapsto [0,1]$ by*

$$\Gamma_\rho(\mu) = Pr[X \le \Phi^{-1}(\mu) \wedge Y \le \Phi^{-1}(\mu)]$$

where X and Y are jointly normal random variables with mean 0 and covariance matrix $\begin{pmatrix} 1 & \rho \\ \rho & 1 \end{pmatrix}$.

The only fact that we will use for stating the stronger form of SSEH is the asymptotic behavior of $\Gamma_\rho(\mu)$ when ρ is close to 1 and μ bounded away from 0.

Fact 4.2.6. *There is a constant $c > 0$ such that for all sufficiently small ϵ and all $\mu \in [1/10, 1/2]$,*[2]

$$\Gamma_{1-\epsilon}(\mu) \le \mu(1 - c\sqrt{\epsilon}).$$

Conjecture 4.2.7 (SSE Hypothesis, Equivalent Formulation [138]). *For every integer $q > 0$ and $\epsilon, \gamma > 0$, it is NP-hard to distinguish between the following two cases for a given regular graph $G = (V, E)$:*

- ***Yes*** *There is a partition of V into q equi-sized sets $S_1, \cdots, S_q$ such that $\Phi_G(S_i) \le 2\epsilon$ for every $1 \le i \le q$.*
- ***No*** *For every $S \subseteq V$, letting $\mu = |S|/|V|$, it holds that $\Phi_G(S) \ge 1 - (\Gamma_{1-\epsilon/2}(\mu) + \gamma)/\mu$,*

[2]Note that the lower bound on μ can be taken arbitrarily close to 0. So the statement holds with $\mu \in [\epsilon', 1/2]$ for any constant $\epsilon' > 0$.

where the $\Gamma_{1-\epsilon/2}(\mu)$ is the Gaussian noise stability.

We present two remarks about the Conjecture 4.2.7 from [9], which are relevant to our proof of Theorem 4.3.5.

Remark 4.2.8. *[9] The* ***Yes*** *instance of Conjecture 4.2.7 implies that the number of edges leaving each S_i is at most $4\epsilon|E|/q$, so the total number of edges not contained in one of the S_i is at most $2\epsilon|E|$.*

Remark 4.2.9. *[9] The* ***No*** *instance of Conjecture 4.2.7 implies that for ϵ sufficiently small, there exists some constant c' such that $\Phi_G(S) \geq c'\sqrt{\epsilon}$, provided that $\mu \in [1/10, 1/2]$ and setting $\gamma \leq \sqrt{\epsilon}$. In particular, $|E(S, V\backslash S)| \geq \Omega(\sqrt{\epsilon}|E|)$, for any $|V|/10 \leq |S| \leq 9|V|/10$.* [3]

Remark 4.2.8 follows from the definition of normalized edge expansion and the fact that sum of degree is two times number of edges. Remark 4.2.9 follows from Fact 4.2.6. The strong form of SSE Hypothesis 4.2.7, Remark 4.2.8, and Remark 4.2.9 will be particularly helpful for proving our SSE-hardness of approximation result (Theorem 4.3.5).

4.2.2 Preliminaries for Corruption Detection on Networks

We model networks as directed or undirected graphs, where each vertex in the network can be one of two types: truthful or corrupted. At times, we will informally call truthful vertices "good" and corrupt vertices "bad." We say that the corrupt party has *budget b* if it can afford to corrupt at most b nodes of the graph. Given a vertex set V, and a budget b, the corrupt entity will choose to control a subset of nodes $\mathcal{B} \subseteq V$ under the constraint $|\mathcal{B}| \leq b$. The rest of the graph remains as truthful vertices, i.e., $\mathcal{T} = V\backslash\mathcal{B} \subseteq V$. We assume that there are more truthful than corrupt nodes ($b < |V|/2$). It is easy to see that in the case where $|\mathcal{B}| \geq |\mathcal{T}|$, the corrupt nodes can prevent the identification of even one truthful node, by simulating truthful nodes (see e.g. [6]).

[3]Recall that Fact 4.2.6 is true for $\mu \in [\epsilon', 1/2]$ for any constant $\epsilon' > 0$. Therefore, Remark 4.2.9 can be strengthened and states, for any $\epsilon'|V| \leq |S| \leq (1-\epsilon')|V|$, $|E(S, V\backslash S)| \geq \Omega(\sqrt{\epsilon}|E|)$. This will be a useful fact for proving hardness of approximation of $m(G, g)$ for finding many truthful nodes in Section 4.5.

Each node audits and reports its (outgoing) neighbors' identities. That is, each vertex $u \in V$ will report the type of each $v \in \mathcal{N}(u)$, which is a vector in $\{0,1\}^{|\mathcal{N}(u)|}$. Truthful nodes always report the truth, i.e., it reports its neighbor $v \in \mathcal{T}$ if v is truthful, $v \in \mathcal{B}$ if v is corrupt. The corrupt nodes report their neighbors' identities adversarially. In summary, a strategy of the bad agents is composed of a strategy to take over at most b nodes on the graph, and reports on the nodes that neighbor them.

Definition 4.2.10 (**Strategy for a corrupt party**). *A strategy for the corrupt party is a function that maps a graph G and budget b to a subset of nodes $\mathcal{B}$ with size $|\mathcal{B}| \leq b$, and a set of reports that each node $v \in \mathcal{B}$ gives about its neighboring nodes, $\mathcal{N}(v)$.*

Definition 4.2.11 (**Computationally bounded corrupt party**). *We say that the corrupt party is computationally bounded if its strategy can only be a polynomial-time computable function.*

The task for the central agency is to find a good node on this corrupted network, based on the reports. It is clear that the more budget the corrupt party has, the harder the task of finding one truthful node becomes. It was observed in [6] that, for any graph, it is not possible to find one good node if $b \geq |V|/2$. If $b = 0$, it is clear that the entire set V is truthful. Therefore, given an arbitrary graph G, there exists a critical number $m(G)$, such that if the bad party has budget lower than $m(G)$, it is always possible to find a good node; if the bad party has budget greater than or equal to $m(G)$, it may not be possible to find a good node. In light of this, we define the critical number of bad nodes on a graph G. First, we formally define what we mean when we say it is impossible to find a truthful node on a graph G.

Definition 4.2.12 (**Impossibility of finding one truthful node**). *Given a graph $G = (V, E)$, the bad party's budget b and reports, we say that it is* impossible to identify one truthful node *if for every $v \in V$ there is a configuration of the identities of the nodes where v is bad, and the configuration is consistent with the given reports, and consists of fewer than or equal to b bad nodes.*

Definition 4.2.13 (**Critical number of bad nodes on a graph** G, $m(G)$)**.** *Given an arbitrary graph $G = (V,E)$, we define $m(G)$ as the minimum number b such that there is a way to distribute b corrupt nodes and set their corresponding reports such that it is impossible to find one truthful node on the graph G, given G, the reports and that the bad party's budget is at most b.*

For example, for a star graph G with $|V| \geq 5$, the critical number of bad nodes is $m(G) = 2$. If there is at most 1 corrupt node on G, the central agency can always find a good node, thus $m(G) \neq 1$. If there are at most 2 bad nodes on G, then the bad party can control the center node and one of the leaves. It is impossible for central agency to find one good node.

Given a graph G, by definition there exists some set of $m(G)$ nodes that can make it impossible to find a good node if they are corrupted. However, this does not mean that the corrupt party can necessarily find this set in polynomial time. Indeed, Theorem 4.3.5 establishes that they cannot always find this set in polynomial time if we assume the SSE Hypothesis (Conjecture 4.2.7) and that P ≠ NP.

4.3 Proofs of Theorems 4.3.2, 4.3.5, and 4.3.11

In the following section, we state our main results by first presenting the close relation of our problem to the k-vertex separator problem. Then we use this characterization to prove Theorem 4.3.2. This characterization will additionally be useful for the proofs of Theorems 4.3.5 and 4.3.11, which we will give in Section 4.3.2 and Section 4.3.3.

4.3.1 2-Approximation by Vertex Separation

Lemma 4.3.1 (2-Approximation by Vertex Separation)**.** *The critical number of corrupt nodes for graph G, $m(G)$, can be bounded by the minimal sum of k-vertex separator and k, $\min_k(S_G(k)+k)$, up to a factor of 2. i.e.,*

$$\frac{1}{2}\min_k (S_G(k)+k) \leq m(G) \leq \min_k (S_G(k)+k)$$

Proof of Lemma 4.3.1. The direction $m(G) \leq \min_k S_G(k) + k$ follows simply. Let $k^* = \arg\min_k (S_G(k) + k)$. If the corrupt party is given $S_G(k^*) + k^*$ nodes to corrupt on the graph, it can first assign $S_G(k^*)$ nodes to the separator, thus the remaining nodes are partitioned into components of size at most k^*. Then it arbitrarily assigns one of the components to be all bad nodes. The bad nodes in the connected components report the nodes in the same component as good, and report any node in the separator as bad. The nodes in the separator can effectively report however they want (e.g. report all neighboring nodes as bad). It is impossible to identify even one single good node, because all connected components of size k can potentially be bad, and all vertices in the separator are bad.

The direction $(1/2)\min_k (S_G(k) + k) \leq m(G)$ can be proved as follows. When there are $b = m(G)$ corrupt nodes distributed optimally in G, it is impossible to find a single good node by definition, and therefore, in particular, the following algorithm (Algorithm 19) cannot always find a good node:

Algorithm 19 Finding one truthful vertex on undirected graph G

Input: Undirected graph G

- If the reports on edge (u, v) does not equal to $(u \in \mathcal{T}, v \in \mathcal{T})$, remove both u, v and any incident edges. Remove a pair of nodes in each round, until there are no bad reports left.
- Call the remaining graph H. Declare the largest component of H as good.

Run Algorithm 19 on G, and suppose the first step terminates in i rounds, then:

- No remaining node reports neighbors as corrupt
- $|V| - 2i$ nodes remain in graph
- $\leq b - i$ bad nodes remain in the graph, because each time we remove an edge with bad report, and one of the end points must be a corrupt vertex.

Note that if two nodes report each other as good, they must be the same type (either both truthful, or both corrupt.) Since graph H only contains good reports, nodes within a connected component of H have the same types. If there exists a component of size larger than $b-i$, it exceeds bad party's budget, and must be all good. Therefore, Algorithm 19 would successfully find a good node.

Since Algorithm 19 cannot find a good node, the bad party must have the budget to corrupt the largest component of H, which means it has size at most $b-i$. Hence, $S_G(b-i) \leq 2i$. Plugging in $b = m(G)$, we get that

$$m(G) = \frac{2i}{2} + b - i \geq \min_k (S_G(k)/2 + k) \geq \frac{1}{2} \min_k (S_G(k) + k),$$

where the first inequality comes from $2i \geq S_G(b-i)$. □

Furthermore, the upperbound in Lemma 4.3.1 additionally tells us that if corrupt party's budget $b \leq m(G)/2$, the set output by Algorithm 19 is guaranteed to be good.

Theorem 4.3.2. *Fix a graph G and suppose that the corrupt party has a budget $b \leq m(G)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without knowledge of either $m(G)$ or b. Furthermore, the central agency's algorithm runs in linear time (in the number of edges in the graph G).*

Proof of Theorem 4.3.2. Suppose the corrupt party has budget $b \leq m(G)/2$. Run Algorithm 19. We remove $2i$ nodes in the first step, and separate the remaining graph H into connected components. Notice each time we remove an edge with bad report, at least one of the end point is a corrupt vertex. So we have removed at most $2b \leq m(G) \leq \lceil |V|/2 \rceil$ nodes. Therefore, the graph H is nonempty, and the nodes in any connected component of H have the same identity. Let $k^* \geq 1$ be the size of the maximum connected component of H. We can conclude that $S_G(k^*) \leq 2i$, since $2i$ is a possible size of k^*-vertex separator of G.

Notice there are at most $b - i \leq m(G)/2 - i$ bad nodes in H by the same fact that at least one bad node is removed each round. By the upper bound in Lemma 4.3.1,

$$b - i \leq m(G)/2 - i \leq \min_k (S_G(k) + k)/2 - i \leq (2i + k^*)/2 - i \leq \frac{k^*}{2}.$$

Since $k^* \geq 1$, the connected component of size k^* exceeds the bad party's remaining budget $k^*/2$, and must be all good.

Algorithm 19 is linear time because it loops over all edges and removes any "bad" edge that does not have reports $(\mathcal{T}, \mathcal{T})$ (takes $\leq |E|$ time when we use a list with "bad" edges at the front), and counts the size of the remaining components ($\leq |V|$ time), and thus is linear in $|E|$. □

Remark 4.3.3. *Both bounds in Lemma 4.3.1 are tight. For the lower bound, consider a complete graph with an even number of nodes. For the upper bound, consider a complete bipartite graph with one side smaller than the other.*

To elaborate on Remark 4.3.3, for the lower bound, in a complete graph with n nodes, the critical number of bad nodes is $n/2$, and $\min_k S_G(k) + k = n$.

For the upper bound, consider a complete bipartite graph $G = (V, E)$. The vertex set is partitioned into two sets $V = S_1 \cup S_2$ where the induced subgraphs on S_1 and S_2 consist of isolated vertices, and every vertex $u \in S_1$ is connected with every vertex $v \in S_2$. The smallest sum of k-vertex separator with k is obtained with $k = 1$, i.e., $\min_k S_G(k) + k = \min\{|S_1|, |S_2|\} + 1$. We argue that this is also the minimal number of bad nodes needed to corrupt the graph. Without loss of generality , let $|S_1| < |S_2|$. If the bad party controls all of S_1 plus one node in S_2, it can prevent the identification of a good node. On the other hand, if the bad party controls $b < |S_1| + 1$ nodes, then we can always identify a good node. Specifically, we are in one of the following cases:

1. The bad party does not control all of S_1. Then there will be a connected component of size $n - b > b$ that report each other as good, because the bad nodes cannot control all of S_2, and any induced subgraph of a complete bipartite graph with nodes on both sides is connected.

2. The bad party controls all of S_1. In this case, the largest connected component of

nodes that all report each other as good is only 1. However, in this case, we conclude that the bad nodes must control all of S_1 and no other node (due to their budget). Hence, any node in S_2 is good.

We end by discussing that the efficient algorithm given in this section does not address the regime when the budget of the bad party, b, falls in $m(G)/2 < b \le m(G)$. Though by definition of $m(G)$, the central agency can find at least one truthful node as long as $b \le m(G)$, by, for example, enumerating all possible assignments of good/bad nodes consistent with the report, and check the intersection of the assignment of good nodes. However, it is not clear that the central agency has a polynomial time algorithm for doing this. Of course, one can always run Algorithm 19, check whether the output set exceeds $b - i/2$, and concludes that the output set is truthful if that is the case. However, there is no guarantee that the output set will be larger than $b - i/2$ if $m(G)/2 < b \le m(G)$. We propose the following conjecture:

Conjecture 4.3.4. *Fix a graph G and suppose that the corrupt party has a budget b such that $m(G)/2 < b \le m(G)$. The problem of finding one truthful node given the graph G, bad party's budget b and the reports is NP-hard.*

4.3.2 SSE-Hardness of Approximation for $m(G)$

In this section, we show the hardness of approximation result for $m(G)$ within any constant factor under the Small Set Expansion (SSE) Hypothesis [137]. Specifically, we prove Theorem 4.3.5.

Theorem 4.3.5. *For every $\beta > 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V, E)$, it is SSE-hard to distinguish between the case where $m(G) \le \epsilon \cdot |V|$ and $m(G) \ge \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes for a graph to within any constant factor is SSE-hard.*

In order to prove Theorem 4.3.5, we construct a reduction similar to [9], and show that the bad party can control auxiliary graph of the **Yes** case of SSE with $b = O(\epsilon|V'|)$ and cannot control the auxiliary graph of the **No** case of SSE with $b = \Omega(\epsilon^{0.51}|V'|)$.

Given an undirected d-regular graph $G = (V, E)$, construct an auxiliary undirected graph $G' = (V', E')$ in the following way [9]. Let $r = d/2$. For each vertex $v^i \in V$, make r copies of v^i and add to the vertex set of G', denoted $v_1^i, \cdots, v_r^i$. Denote the resulting set of vertices as $\tilde{V} = V \times \{1, \cdots, r\}$. Each edge $e^k \in E$ of G becomes a vertex in G', denoted e^k. Denote this set of vertices as $\tilde{E}$. In other words, $V' = \tilde{V} \cup \tilde{E} = V \times \{1, \cdots, r\} \cup E$. There exists an edge between a vertex v_j^i and a vertex e^k of G' if v^i and e^k were adjacent edge and vertex pair in G. Note that G' is a bipartite d-regular graph with $d/2|V| + |E| = 2|E|$ vertices.

Lemma 4.3.6. *Suppose $q = 1/\epsilon$, and G can be partitioned into q equi-sized sets $S_1, \cdots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$. Then the bad party can control the auxiliary graph G' with at most $4\epsilon|E| = 2\epsilon|V'|$ nodes.*

Proof of Lemma 4.3.6. Notice by Remark 4.2.8, the total number of edges in G not contained in one of the S_i is at most $2\epsilon|E|$.

This implies that a strategy for the bad party to control graph G' is as follows. Control vertex $e^k \in \tilde{E}$ if $e^k \in E$ is not contained in any of the S_is in G. Call the set of such vertices $E^* \subseteq \tilde{E}$. Let $S_i^* \subseteq V'$ be the set that contains all r copies of nodes in $S_i \subseteq V$. Control one of the S_i^*s, say S_1^*. Control all the edge nodes in $\tilde{E}$ that are adjacent to S_1^*. Call this set $\mathcal{N}(S_1^*)$. The corrupt nodes in $S_1^* \cup \mathcal{N}(S_1^*)$ report their neighbors in $S_1^* \cup \mathcal{N}(S_1^*)$ as good, and report E^* as bad. Nodes in E^* can effectively report however they want; suppose they report every neighboring node as bad. Then, it is impossible to identify even one truthful node, since assigning any S_i^* as corrupt is consistent with the report and within bad party's budget.

This strategy controls $|E^*| + |S_i^*| + |\mathcal{N}(S_1^*) \setminus E^*|$ nodes on G'. Note that $|\mathcal{N}(S_1^*) \setminus E^*|$ is equal to the number of edges that are totally contained in S_1 on G, which is bounded by $|S_1| \cdot d/2$ (that is if all edges adjacent to S_1 are totally contained in S_1). If $q = 1/\epsilon$, this strategy amounts to controlling $|E^*| + |S_i^*| + |\mathcal{N}(S_1^*) \setminus E^*| \leq 2\epsilon|E| + d/2 \cdot |V|/q + |V|/q \cdot d/2 = 4\epsilon|E| = 2\epsilon|V'|$ nodes on G'. Notice, this number is guaranteed to be smaller than $1/2|V'|$, as long as $q > 4$.

□

Note that, different from the argument in [9], we cannot take r to be arbitrarily large (e.g. $> O(|V||E|)$). This is because when r is large, $2\epsilon|E| + r\cdot|V|/q = O(\epsilon(|E|+|V'|)) = O(\epsilon|V'|)$, and will not be comparable with the $O(\sqrt{\epsilon}|E|)$ in Lemma 4.3.7.

Lemma 4.3.7. *Let $G = (V,E)$ be an undirected d-regular graph with the property that for every $|V|/10 \le |S| \le 9|V|/10$ we have $|E(S, V \setminus S)| \ge \Omega(\sqrt{\epsilon}|E|)$. If bad party controls $O(\epsilon^{0.51}|E|) = O(\epsilon^{0.51}|V'|) < 1/2|V'|$ nodes on the auxiliary graph G' constructed from G, we can always find a truthful node on G'.*

Proof of Lemma 4.3.7. Assume towards contradiction that the bad party controls $O(\epsilon^{0.51}|E|)$ vertices of graph G', and we can't identify a truthful node.

Claim 4.3.8. *If the bad party controls $O(\epsilon^{0.51}|E|)$ vertices of graph G', and it is impossible to identify a truthful node, then there exists a set C of size $O(\epsilon^{0.51}|E|)$ and separates $V'\backslash C$ into sets $\{\mathcal{T}_i'\}_{i=1,\cdots,\ell}$, each of size $O(\epsilon^{0.51}|E|)$.*

Proof of Claim 4.3.8. Since the bad nodes can control G' with $O(\epsilon^{0.51}|E|)$ vertices, $m(G') \le O(\epsilon^{0.51}|E|)$. By the lower bound in Lemma 4.3.1, $\min_k(S_{G'}(k)+k) \le 2m(G') \le O(\epsilon^{0.51}|E|)$. Let $k^* = \arg\min_k(S_{G'}(k)+k)$. Then $k^* \le O(\epsilon^{0.51}|E|)$, $S_{G'}(k^*) \le O(\epsilon^{0.51}|E|)$. By definition of $S_{G'}(k^*)$, there exists a set of size $S_{G'}(k^*)$ whose removal separates the remainder of the graph G' to connected components of size at most k^*. □

Let C and $\mathcal{T}_i'$ be the sets guaranteed by Claim 4.3.8. Note we have taken $r = d/2$, and thus $|\tilde{V}| = |\tilde{E}|$. In other words, half of the V' are "vertex" vertices $\tilde{V}$, and half are "edge" vertices $\tilde{E}$. Therefore, with sufficiently small ϵ, $|C \cap \tilde{V}| \le |C| < 1/2|\tilde{V}|$, $|(\cup_{i=1}^{\ell}\mathcal{T}_i')\cap\tilde{V}| \ge 1/2|\tilde{V}|$, $|\mathcal{T}_i'\cap\tilde{V}| \le |\mathcal{T}_i'| < 3/10|\tilde{V}|$ for every i. Therefore, we can merge the different $\mathcal{T}_i'$'s in Claim 4.3.8, and have two sets $\mathcal{T}_1'$ and $\mathcal{T}_2'$, such that $|\mathcal{T}_1' \cap \tilde{V}| \ge |\tilde{V}|/5$ and $|\mathcal{T}_2' \cap \tilde{V}| \ge |\tilde{V}|/5$. Furthermore, $\mathcal{T}_1'$ and $\mathcal{T}_2'$ are disjoint, and $\mathcal{T}_1', \mathcal{T}_2'$, and C cover V'.

Similar to the proof of Lemma 5.1 in [9], we let $\mathcal{T}_1 \subseteq V$ (resp. $\mathcal{T}_2 \subseteq V$) be the set of vertices $v \in V$ such that some copy of v appears in $\mathcal{T}_1'$ (resp. $\mathcal{T}_2'$). Let $S \subseteq V$ be the set of vertices $v \in V$ such that *all* copies of v appear in C. Since $|\mathcal{T}_1' \cap \tilde{V}|, |\mathcal{T}_2' \cap \tilde{V}| \ge |\tilde{V}|/5 = r|V|/5$, both $|\mathcal{T}_1|, |\mathcal{T}_2| \ge |V|/5$. Furthermore, we observe that $\mathcal{T}_1 \cup \mathcal{T}_2 \cup S = V$, which follows since $\mathcal{T}_1' \cup \mathcal{T}_2' \cup C = V'$. Now we can lower bound $|\mathcal{T}_1 \cup \mathcal{T}_2|$ as follows.

$$|\mathcal{T}_1 \cup \mathcal{T}_2| = |V \setminus S| \geq |V| - |C|/r \geq |V| - c\epsilon^{0.51}|E|/r = |V| - c\epsilon^{0.51}|V|,$$

where the first equality uses the fact that $\mathcal{T}_1 \cup \mathcal{T}_2 \cup S = V$ and that $\mathcal{T}_1 \cup \mathcal{T}_2$ is disjoint from S, and the following inequality uses the fact that $|S| \leq |C|/r$, which follows by definition.

Since $|\mathcal{T}_1 \cup \mathcal{T}_2|$ is sufficiently large, we can find a balanced partition of $\mathcal{T}_1 \cup \mathcal{T}_2$ into sets $S_1 \subseteq \mathcal{T}_1$, $S_2 \subseteq \mathcal{T}_2$, such that $S_1 \cap S_2 = \emptyset, S_1 \cup S_2 = \mathcal{T}_1 \cup \mathcal{T}_2$, and $|V|/10 \leq |S_1|, |S_2| \leq 9|V|/10$. From the property of G that $E(S, V \setminus S) \geq \Omega(\sqrt{\epsilon}|E|)$ in Lemma 4.3.7 and the fact that G is d-regular, we know that

$$E(S_1, S_2) = E(S_1, V \setminus S_1) - E(S_1, S) \geq \alpha\sqrt{\epsilon}|E| - d(\epsilon^{0.51}|E|/r) = \alpha\sqrt{\epsilon}|E| - 2\epsilon^{0.51}|E| = \Omega(\sqrt{\epsilon}|E|),$$

for some constant α. In the first equality we use the fact that S_1, S_2, S form a partition of V. Thus $E(S_1, V \setminus S_1) = E(S_1, S_2 \cup S) = E(S_1, S_2) + E(S_1, S)$.

Note that since $S_1 \subseteq \mathcal{T}_1$ and $S_2 \subseteq \mathcal{T}_2$, and $\mathcal{T}_1'$ and $\mathcal{T}_2'$ do not have edge between them in G', the edges $E(S_1, S_2)$ all have to land as "edge vertices" in C. In other words, for any $u \in S_1$, and $v \in S_2$, if $(u, v) \in E$, then the vertex $(u, v) \in V'$ has to be included in the set C, thus $|C| \geq \Omega(\sqrt{\epsilon}|E|)$.

This contradicts the fact that there are only $O(\epsilon^{0.51}|E|)$ vertices in C.

□

Combining Lemma 4.3.6 and Lemma 4.3.7, Theorem 4.3.5 follows in standard fashion. We give a proof here for completeness.

Proof of Theorem 4.3.5. Suppose for contradiction that there exists some constant $\beta > 0$ such that there is polynomial time algorithm $\mathcal{A}$ that does the following. For any $\epsilon' > 0$ and an arbitrary graph $G' = (V', E')$, it can distinguish between the case where $m(G') \leq \epsilon' \cdot |V'|$ and $m(G') \geq \beta \cdot \epsilon' \cdot |V'|$. Specifically, we will suppose this holds for $\epsilon' < \frac{1}{\beta^{2.05}}$. Then we can use this algorithm to decide the SSE problem as follows.

Fix $\epsilon < \frac{1}{1.5\beta^{2.05}}$, $q = 1/\epsilon$, $\gamma > 0$ sufficiently small ($\leq o(\sqrt{\epsilon})$ suffices). Let $G = (V, E)$ be an arbitrary input to the resulting instance of the SSE decision problem (from Conjecture 4.2.7). Construct the graph $G' = (V', E')$ from G as done in the beginning of Section 4.3.2.

If G was from the YES case of Conjecture 4.2.7, then $m(G') \leq 1.5\epsilon|V'|$ (Lemma 4.3.6). If G was from the NO case of Conjecture 4.2.7, then $m(G') > \epsilon^{0.51}|V'|$ (Lemma 4.3.7). We can invoke our algorithm $\mathcal{A}$ to distinguish these two cases, by letting $\epsilon' = 1.5\epsilon$ and noting that $\beta < (1/(\epsilon')^{0.49})$ by design, which would decide the problem in Conjecture 4.2.7 in polynomial time. □

Now, we can obtain the following Corollary 4.3.9 from Theorem 4.3.5.

Corollary 4.3.9. *Assume the SSE Hypothesis and that P ≠ NP. Fix any $\beta > 1$. There does not exist a polynomial-time algorithm that takes as input an arbitrary graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\beta \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

In summary, the analysis in this section tells us that given an arbitrary graph, it is hard for bad party to corrupt the graph with minimal resources. On the other hand, if the budget of bad nodes is a factor of two less than $m(G)$, a good party can always be detected with an efficient algorithm, e.g. using Algorithm 19.

4.3.3 An $O(\log|V|)$ Approximation Algorithm for $m(G)$

In light of the SSE-hardness of approximation of $m(G)$ within any constant, and the close relation of $m(G)$ with k-vertex separator, we leverage the best known approximation result for k-vertex separator to propose an $O(\log n)$ approximation algorithm for $m(G)$. It is useful as a test for central authorities for measuring how corruptible a graph is. Notably, it is also a potential algorithm for (computationally restricted) bad party to use to decide which nodes to corrupt.

The paper [105] presents an bicritera approximation algorithm for k-vertex separator, with the guarantee that for each k, the algorithm finds a subset $A_k \subseteq V$ such that $|A_k| \leq$

$O(\frac{\log k}{\epsilon}) \cdot S_G(k)$, and the induced subgraph $G_{V \backslash A_k}$ is divided into connected components each of size at most $k/(1-2\epsilon)$ vertices.

Proposition 4.3.10 (Theorem 1.1, [105]). *For any $\epsilon \in (0, 1/2)$, there is a polynomial-time $(\frac{1}{1-2\epsilon}, O(\frac{\log k}{\epsilon}))$- bicriteria approximation algorithm for k-vertex separator.*

Interested readers can refer to [105] Section 3 for the description of the algorithm. Leveraging this algorithm for k-vertex separator, we can obtain a polynomial-time algorithm for seeding corrupt nodes and preventing the identification of a truthful node.

Theorem 4.3.11 ($O(\log|V|)$ Approximation Algorithm). *There is a polynomial-time algorithm that takes as input a graph $G = (V, E)$ and outputs a set of nodes S with size $|S| \leq O(\log|V| \cdot m(G))$, such that corrupting S prevents the central agency from finding a truthful node.*

Proof. The algorithm is as follows. Call the bicriteria algorithm for approximating k-vertex separator in [105] n times, once for each k in $k = 1, \cdots, n$, where $n = |V|$. Each time the algorithm outputs a set of vertices A_k that divides the remaining graph into connected components with maximum size $g(k)$. Choose the k^* for which the algorithm outputs the smallest value of $\min_k |A_k| + g(k)$. The bad party can control A_{k^*} and one of the remaining connected components (the size of which is at most $g(k^*)$), and be sure to prevent the identification of one good node, by the same argument that lead to the upper bound in Lemma 4.3.1.

We now prove that $|A_{k^*}| + g(k^*)$ is an $O(\log|V|)$ approximation for the quantity of consideration $\min_k S_G(k) + k$. For each k, we denote our approximation for $S_G(k) + k$ as $f(k) := |A_k| + g(k)$. Then by the guarantee given in Proposition 4.3.10, we know

$$f(k) = |A_k| + g(k) \leq O\left(\frac{\log k}{\epsilon}\right) \cdot S_G(k) + \frac{1}{1-2\epsilon} k \leq O\left(\frac{\log k}{\epsilon}\right) \cdot (S_G(k) + k).$$

Thus

$$\min_k f(k) \leq \min_k O\left(\frac{\log k}{\epsilon}\right) \cdot (S_G(k) + k) \leq O\left(\frac{\log n}{\epsilon} \min_k (S_G(k) + k)\right) \leq O\left(\log n \cdot m(G)\right).$$

The last inequality follows from the fact that that $\min_k(S_G(k)+k)/2 \leq m(G) \leq \min_k(S_G(k)+k)$ in Lemma 4.3.1, and by taking ϵ to be a fixed constant, e.g. $\epsilon = 1/3$. So $\min_k f(k)$ provides an $O(\log n)$ approximation of $m(G)$. The algorithm consists of n calls of the polynomial-time algorithm in Proposition 4.3.10, so is also polynomial-time. □

4.4 Directed Graphs

Here we present the variant of our problem on directed graphs. As discussed in [6], this is motivated by the fact that in various auditing situations, it may not be natural that any u will be able to inspect v whenever v inspects u.

Given a directed graph $D = (V, E_D)$, we are asked to to find $m(D)$, the minimal number of corrupted agents needed to prevent the identification of a single truthful agent. Firstly, since undirected graphs are special cases of directed graphs, it is clear that the worst case hardness of approximation results still hold. In this section, we will define a analogous notion of vertex separator relevant to corruption detection for directed graphs, and state the version of Theorem 4.3.2 for directed graphs.

Definition 4.4.1 (**Reachability Index**). *On a directed graph $D = (V, E_D)$, say a vertex s can reach a vertex t if there exists a sequence of adjacent vertices (i.e. a path) which starts with s and ends with t. Let $R_D(v)$ be the set of vertices that can reach a vertex v. Define the* ***reachability index*** *of v as $|R_D(v)|$, or in other words, as the total number of nodes that can reach v.*

Based on the notion of reachability index, we design the following algorithm, Algorithm 20, for detecting one good node on directed graphs:

Algorithm 20 Finding one truthful vertex on directed graph D

Input: Directed graph D

- If node u reports node v as corrupt, remove both u, v and any incident edges (incoming and outgoing). Remove a pair of nodes in each round. Continue until there are no bad reports left.
- Call the remaining graph $H = (V_H, E_H)$. Declare a vertex in H with maximum reachability index as good.

Run Algorithm 20 on directed graph D, and suppose the first step terminates in i rounds. Then:

- No remaining node reports out-neighbors as corrupt
- $|V| - 2i$ nodes remain in graph
- $\leq b - i$ bad nodes remain in the graph, because each round in step 1 removes at least one bad node.

The main idea is that, if there exists a node v with reachability index larger than $b - i$, at least $b - i$ nodes claim (possibly indirectly) that v is good, which means at least one good node also reports v as good, and thus v must be good. In the rest of the section, we use this observation to generalize Theorem 4.3.2.

We define a notion similar to k-vertex separator on directed graphs, show that our notion provides a 2-approximation for $m(D)$ when D is a directed graph, and that the equivalent of Theorem 4.3.2 also holds in the directed case.

Definition 4.4.2 (**k-reachability separator**). *We say a set of vertices $S \subseteq V$ is a k-reachability separator of a directed graph $D = (V, E_D)$ if after the removal of S and any adjacent edges, all vertices in the remaining graph are of reachability at most k.*

Since in an undirected graph, any pair of vertices can reach each other if and only if they belong to the same connected component, one can check that k-reachability separator on an undirected graph is exactly equivalent to a k-vertex separator. Thus we use a similar notation, $S_D(k)$, to denote the size of the minimal k-reachability separator on D.

Lemma 4.4.3 (2-Approximation Lemma on Directed Graphs)**.**

$$\frac{1}{2}\min_k(S_D(k)+k) \le m(D) \le \min_k(S_D(k)+k)$$

Proof. The direction $m(D) \le \min_k S_D(k)+k$ is proved as follows. Let $k^* = \arg\min_k(S_D(k)+k)$. If the corrupt party is given $\min_k(S_D(k)+k)$ nodes to allocate on D, it can first assign $S_D(k^*)$ nodes to a k^*-reachability separator C, such that the remaining nodes have reachability index at most k^*. Then it arbitrarily assigns one of the vertices v^* with maximum reachability index plus its $R_H(v^*)$ as bad. The bad nodes in $R_H(v^*)$ report any neighbor in the separator C as bad and any other neighbor as good. The nodes in the separator can effectively report however they want (e.g. report all neighboring nodes as bad).

It is impossible to detect a single good node, because every node v can only be reached by $R_H(v)$ and C. For every $v \in H$, it being assigned as corrupt or good is consistent with the reports. If v is corrupt, $R_H(v)$ is also assigned as corrupt, thus all nodes in H receive good reports from $R_H(v)$, bad reports from C and give bad reports to C. If v is truthful, all nodes still receive and give the same reports. So for every $v \in V_H$, assigning $R_H(v)$ as bad, and $V_H \setminus R_H(v)$ as good is consistent with the observed reports. It is impossible to find a good node in H by definition.

The proof for $1/2\min_k(S_D(k)+k) \le m(D)$ is given by Algorithm 20. Let there be $m(D)$ bad nodes distributed optimally on the graph. By definition, these nodes prevent the identification of a good node. Run Algorithm 20, and suppose the first step terminates in i rounds. This means we have removed at least i bad nodes, and there are at most $m(D)-i$ bad nodes left on H. If there exists a node on H with reachability $m(D)-i$, then this node must be truthful, since there are not enough bad nodes left to corrupt all the nodes that can reach it, and all the reports in the remaining graph are good. Thus $|R(v)| < m(D)-i$ for any v. Therefore, the set of $2i$ removed nodes must be an $m(D)-i$ reachability separator. Hence, we can bound $m(D)$ as follows.

$$m(D) = (m(D) - i) + 2i/2 \geq \min_k(k + S_D(k)/2) \geq \frac{1}{2}\min_k(S_D(k) + k)$$

where the first inequality follows from the fact that $2i \geq S_D(m(D) - i)$. □

Theorem 4.4.4. *Fix a directed graph D and suppose that the corrupt party has a budget $b \leq m(D)/2$. Then the central agency can identify a truthful node, regardless of the strategy of the corrupt party, and without the knowledge of either $m(D)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

Proof of Theorem 4.4.4. Suppose the corrupt party has budget $b \leq m(D)/2$. Run Algorithm 20. Notice each time we remove an edge with bad report, at least one of the end point is a corrupt vertex. So we have removed at most $2b \leq m(D) \leq \lceil |V|/2 \rceil$ nodes. Therefore, the graph H is nonempty. Let $k^* \geq 1$ be the maximum reachability index in H. Since $b \leq m(D)/2$, and there are no bad reports in H, the reachability index of a bad node in graph H is at most $m(D)/2 - i \leq \min_k(S_D(k) + k)/2 - i \leq (2i + k^*)/2 - i = k^*/2 < k^*$.

Then a vertex with reachability index k^* must be found by Algorithm 20, and must be a truthful node. The linear runtime $O(|E_D|)$ follows from the same analysis as in the proof of Theorem 4.3.2. □

4.5 Finding an Arbitrary Fraction of Good Nodes on a Graph

Being able to detect one good node may seem limited, but in fact, the same arguments and construction can be adapted to show that approximating the critical number of bad nodes to prevent detection of any arbitrary δ fraction of good nodes is SSE-hard. In this section, we propose the definition of g-remainder k-vertex separator, a vertex separator notion related to identifying arbitrary number of good nodes, present a 2-approximation result, and prove hardness of approximation with arguments similar to proof of Theorem 4.3.5 in Section 4.3.2.

We abuse notation and define $m(G, g)$ to be the minimal number of bad nodes needed to prevent the identification of g nodes.

Definition 4.5.1 ($m(G,g)$)**.** *We define $m(G,g)$ as the minimal number of bad nodes such that it is impossible to find g good nodes in G. In particular, $m(G) = m(G,1)$.*

Definition 4.5.2 (g**-remainder** k**-vertex Separator**)**.** *Consider the following separation property: after the removal of a vertex set S, the remaining graph $G_{V\setminus S}$ is a union of connected components, where connected components of size larger than k sum up to size less than g. We call such a set S a g-remainder k-vertex separator of G.*

For any integer $0 < k, g < |V|$, we denote the minimal size of such a set as $S_G(k,g)$. In particular, a minimal k-vertex separator is a 1-remainder k-vertex separator, i.e., $S_G(k) = S_G(k,1)$.

Theorem 4.5.3. *Fix a graph G and the number of good nodes to recover, g. Suppose that the corrupt party has a budget $b \leq m(G,g)/2$. If $g < |V| - 2b$, then the central agency can identify g truthful nodes, regardless of the strategy of the corrupt party, and without knowledge either of $m(G,g)$ or b. Furthermore, the central agency's algorithm runs in linear time.*

Algorithm 21 Finding g truthful vertices on an undirected graph G

Input: Undirected graph G

```
• If the reports on edge $(u,v)$ does not equal to $(u \in \mathcal{T}, v \in \mathcal{T})$, remove both $u,v$ and
  any incident edges. Remove a pair of nodes in each round, until there are no bad
  reports left.
• Suppose the previous step terminates in $i$ rounds. In the remaining graph $H$, rank
  the connected component from large to small by size. Declare the largest component
  as good and remove the declared component until we have declared $g$ nodes as good.
```

Proof of Theorem 4.5.3. We claim that central agency can use Algorithm 21, and output at least g good nodes if $b \leq m(G,g)/2$. Step 1 of Algorithm 19 must terminate after removing fewer than $m(G,g)$ nodes, because each round has to remove at least one bad node, and there are only $m(G,g)/2$ bad nodes in total. Let the number of nodes removed be $m(G,g)-\delta$, so at least $m(G,g)/2 - \delta/2 \geq b - \delta/2$ are corrupt. Thus at most $\delta/2$ bad nodes remain in the graph H.

Assume towards contradiction that only $y < g$ nodes output by Algorithm 19 are good. This means that the $m(G,g) - \delta$ removed nodes separate the graph G into connected components where all components with size larger than $\delta/2$ sum to fewer than g. Then $m(G,g) - \delta = m(G,y)$ for $y < g$, contradicting the fact that $m(G,g)$ is the minimum budget needed to prevent identification of g nodes.

□

In fact, just like in Section 4.3, Algorithm 21 additionally gives us a characterization of $m(G,g)$ in terms of the size of the smallest g-remainder k-vertex separator of a graph, for an appropriately chosen value of k.

Lemma 4.5.4 (2-Approximation by Vertex Separation). *The minimal sum of g-remainder k-vertex separator and k, $\min_k (S_G(k,g) + k)$, bounds the critical number of bad nodes $m(G,g)$ up to a factor of 2. i.e.,*

$$\frac{1}{2} \min_k S_G(k,g) + k \leq m(G,g) \leq \min_k S_G(k,g) + k.$$

Proof of Lemma 4.5.4. The upper bound follows simply. Let $k^* = \arg\min_k S_G(k,g)+k$. Given a budget $b = \min_k S_G(k,g) + k$, the bad party can remove a set of size $S_G(k^*,g)$ and separate the graph into connected components of size at most k^*, except for fewer than g nodes. Control one of the connected components of size at most k^*, and construct the reports similarly as in Lemma 4.3.1. Then the central agency can only identify fewer than g good nodes.

For the lower bound, suppose there are $b = m(G,g)$ bad nodes distributed optimally on G and thus it's impossible to find g good nodes by definition. Run Algorithm 21. Suppose the first step terminates in i rounds. After the removal of $2i$ nodes, the graph must be separated into connected components smaller than $b - i$, except for fewer than g nodes. Then $2i \geq S_G(b - i, g)$. Therefore,

$$\frac{1}{2}\min_k (S_G(k,g) + k) \leq \min_k \left(\frac{S_G(k,g)}{2} + k \right) \leq \frac{1}{2} S_G(b-i,g) + (b-i) \leq \frac{2i}{2} + b - i = m(G,g)$$

$\square$

Now using the characterization given by g-remainder k-vertex separator, we are ready to prove that it is SSE-hard to approximate the budget needed to prevent any arbitrary number of good nodes, i.e., $m(G,g)$ for any $g < |V|/3$.

Theorem 4.5.5. *For every $\beta > 1$ and every $0 < \delta < 1$, there is a constant $\epsilon > 0$ such that the following is true. Given a graph $G = (V,E)$, it is SSE-hard to distinguish between the case where $m(G, \delta|V|) \leq \epsilon \cdot |V|$ and $m(G, \delta|V|) \geq \beta \cdot \epsilon \cdot |V|$. Or in other words, the problem of approximating the critical number of corrupt nodes such that it is impossible to find $\delta|V|$ good nodes within any constant factor is SSE-hard.*

We first prove Theorem 4.5.5 for $0 < \delta < 1/3$. The proof in this regime follows similar constructions and arguments as in the proof of Theorem 4.3.5. Note that the proof extends naturally for any $0 < \delta < 1/2$. This is effectively because the range for μ in Remark 4.2.9 can be made to $[\epsilon', 1/2]$, for any constant $\epsilon' > 0$. Further explanation is provided in proof for Lemma 4.5.7.

Firstly, we construct G' based on G as in Section 4.3.2. Lemma 4.3.6 immediately implies that:

Lemma 4.5.6. *Suppose $q = 1/\epsilon$, and G can be partitioned into q equi-sized sets $S_1, \cdots, S_q$ such that $\Phi_G(S_i) \leq 2\epsilon$ for every $1 \leq i \leq q$. The bad party can prevent the identification of one good node, and thus $\delta|V'|$ good nodes, on the auxiliary graph G' with $O(\epsilon|E|) = O(\epsilon|V'|)$ nodes.*

We reprove the analogous lemma to Lemma 4.3.7.

Lemma 4.5.7. *Let $G = (V,E)$ be an undirected d-regular graph with the property that for every $|V|/10 \leq |S| \leq 9|V|/10$ we have $|E(S, V\ S)| \geq \Omega(\sqrt{\epsilon}|E|)$. If bad party controls $O(\epsilon^{0.51}|E|) = O(\epsilon^{0.51}|V'|) < 1/2|V'|$ nodes on the auxiliary graph G' constructed from G, we can always find $\delta|V'|$ truthful nodes on G', for any $\delta < 1/3$.*

Proof of Lemma 4.5.7. Let $g = \delta|V'|$. Assume towards contradiction that the bad party controls $O(\epsilon^{0.51}|E|)$ vertices in G', and we cannot identify g truthful nodes.

Claim 4.5.8. *If the bad party controls $O(\epsilon^{0.51}|E|)$ vertices of graph G', and we can't identify g truthful node, then there exists a set C of size $O(\epsilon^{0.51}|E|)$ and separates $V'\backslash C$ into sets $\{\mathcal{T}_i'\}_{i=1,\cdots,\ell}$, each of size $|\mathcal{T}_i'| \leq O(\epsilon^{0.51}|E|)$, and sets $\{A_j'\}_{j=1,\cdots,K}$, each of size $|A_j'| > \Omega(\epsilon^{0.51}|E|)$, and $|\cup_j^K A_j'| < g$.*

Proof of Claim 4.5.8. Since the corrupt party can control G' with $O(\epsilon^{0.51}|E|)$ vertices, $m(G',g) \leq O(\epsilon^{0.51}|E|)$. By Lemma 4.5.4 $\min_k S_{G'}(k,g)+k \leq 2m(G',g) \leq O(\epsilon^{0.51}|E|)$. Let $k^* = \arg\min_k S_{G'}(k,g)+k$. Then $k^* \leq O(\epsilon^{0.51}|E|)$, $S_{G'}(k^*,g) \leq O(\epsilon^{0.51}|E|)$. By definition of $S_{G'}(k^*,g)$, there exists a set of size $S_G(k^*)$ after whose removal separates the remainder of the graph G to connected components of size at most k^* except for fewer than g nodes. Thus components of size larger than $\Omega(\epsilon^{0.51}|E|)$ contain fewer than g nodes. □

Let $\mathcal{T}' = \cup_{i=1}^{\ell}\mathcal{T}_i', A' = \cup_{j=1}^{K}A_j'$. Since $|C| = O(\epsilon^{0.51}|E|) = O(\epsilon^{0.51}|\tilde{V}|)$, and $C \cup \mathcal{T}' \cup A' = V'$, for small enough ϵ, $|(\mathcal{T}' \cup A') \cap \tilde{V}| \geq 9|\tilde{V}|/10$. From the assumption that we can't identify g truthful nodes, $|A'| < g \leq |V'|/3$. Otherwise, we can claim the entire A' as good and identify g truthful nodes. Thus $|A' \cap \tilde{V}| \leq |V'|/3 \leq 2/3|\tilde{V}|$.[4]

Additionally, use the fact that $|\mathcal{T}_i' \cap \tilde{V}| < |\tilde{V}|/10$ for every i, with sufficiently small ϵ, we can merge various sets in $\{\{A_j\}_{j=1,\cdots,K}, \{\mathcal{T}_i\}_{i=1,\cdots,\ell}\}$ and get two sets V_1' and V_2', such that $|V_1' \cap \tilde{V}|, |V_2' \cap \tilde{V}| \geq |\tilde{V}|/10$, and V_1' and V_2' are separated by C.

Now, let $V_1 \subseteq V$ (resp. $V_2 \subseteq V$) be the set of vertices $v \in V$ such that some copy of v appears in V_1' (resp. V_2'). Let $S \subseteq V$ be the set of vertices $v \in V$ such that all r copies of v

[4]If we use the fact that $|A'| < g \leq (|V'| - \epsilon'|V|)/2$, for some constant ϵ', then $|A' \cap \tilde{V}| \leq (|V'| - \epsilon'|V'|)/2 \leq (1-\epsilon')|\tilde{V}|$. We can merge $\{\{A_j\}, \{\mathcal{T}_i\}\}$ to two sets V_1', V_2' such that $|V_1' \cap \tilde{V}|, |V_2' \cap \tilde{V}| \geq \epsilon'|\tilde{V}|$. The rest of the proof still goes through.

appears in C. Since $|V_1' \cap \tilde{V}|, |V_2' \cup \tilde{V}| \geq |\tilde{V}|/10 = r|V|/10$, both $|V_1|, |V_2| \geq |V|/10$. Furthermore, we observe that $V_1 \cup V_2 \cup S = V$, which follows from $V_1' \cup V_2' \cup C = V'$. Now we can lower bound $|V_1 \cup V_2|$ as follows.

$$|V_1 \cup V_2| = |V \backslash S| \geq |V| - |C|/r \geq |V| - c\epsilon^{0.51}|E|/r = |V| - c\epsilon^{0.51}|V|$$

The first equality again follows from the fact that $V_1 \cup V_2 \cup S = V$, and that $V_1 \cup V_2$ is disjoint from S, and the second inequality follows by definition of S.

Since $V_1 \cup V_2$ is sufficiently large, we can find a balanced partition of $V_1 \cup V_2$ into sets $S_1 \subseteq V_1$, $S_2 \subseteq V_2$, $S_1 \cap S_2 = \varnothing, S_1 \cup S_2 = V_1 \cup V_2$, $|V|/10 \leq |S_1|, |S_2| \leq 9|V|/10$. From the property of G that $E(S, V \setminus S) \geq \Omega(\sqrt{\epsilon}|E|)$ in Lemma 4.3.7 and the fact that G is d-regular, we know that

$$E(S_1, S_2) = E(S_1, V \setminus S_1) - E(S_1, S) \geq \alpha\sqrt{\epsilon}|E| - d(\epsilon^{0.51}|E|/r) = \alpha\sqrt{\epsilon}|E| - 2\epsilon^{0.51}|E| = \Omega(\sqrt{\epsilon}|E|),$$

for some constant α. In the first equality, we use the fact that $S_1 \cup S_2 \cup S = V$, and S_1, S_2, S are disjoint. Thus $E(S_1, V \backslash S_1) = E(S_1, S_2 \cup S) = E(S_1, S_2) + E(S_1, S)$.

Note that since $S_1 \subseteq V_1$ and $S_2 \subseteq V_2$, and $\mathcal{T}_1'$ and $\mathcal{T}_2'$ do not have edges between them in G', the edges $E(S_1, S_2)$ all have to land as "edge vertices" in C. Formally, $E(S_1, S_2) \subseteq \tilde{E} \cap C$. In other words, for any $u \in S_1$, and $v \in S_2$, if $(u, v) \in E$, then the vertex $(u, v) \in V'$ has to be included in the set C, thus $|C| \geq \Omega(\sqrt{\epsilon}|E|)$.

This contradicts the fact that there are only $O(\epsilon^{0.51}|E|)$ vertices in C. □

Using Lemma 4.5.6 and Lemma 4.5.7, we can again obtain Theorem 4.5.5 for $0 < \delta < 1/2$, with the same argument for the proof of Theorem 4.3.5 in Section 4.3.2.

When $1/2 \leq \delta < 1$, we construct an auxiliary graph in the following way. Take as input any graph $G = (V, E)$. Let $h = \delta/(1-\delta)|V|$, construct $G' = G \cup h$-clique. Note $h = \delta|V'|$. Then, we claim that the critical number of bad nodes such that it is impossible to detect $\delta|V'| + 1$ good nodes on G' is the same as the critical number of bad nodes such that it is impossible to find one good node on G.

Claim 4.5.9. *Given any graph G, $1/2 \leq \delta < 1$ and G' as constructed,*

$$m(G', \delta|V'| + 1) = m(G).$$

Proof. Firstly, observe that

$$\delta|V'| = \delta(|V| + h) = \delta(|V| + \delta/(1-\delta)|V|) = h.$$

Therefore, one way to prevent identification of $\delta|V'|+1$ good nodes on G' is to prevent identification of one good node on G. Since the h-clique is of size at least $|V'|/2$, and report each other as good, they will be detected as good nodes. This strategy requires bad party to have budget $b = m(G)$. Thus $m(G', \delta|V'|+1) \leq m(G)$.

The direction $m(G', \delta|V'|+1) \geq m(G)$ follows by the fact that the strategy above is optimal. In order to prove this, we make the following observation:

Claim 4.5.10. *Given any graph G and $g \leq |V|$,*

$$m(G) \leq m(G, g) + g - 1$$

Proof of Claim 4.5.10. One way to prevent identification of one good node is to corrupt $m(G, g)$ nodes plus the (at most) $g-1$ detected good nodes. Call the set of the $g-1$ or fewer detected nodes S. Notice that any node in $G \backslash S$ that is adjacent to S are reported as bad by S. If not, this node has the same identity with S, and should be detected as good as well. Therefore, the bad party is able to corrupt the set S without incurring any change in the reports, since all edges incident to S now have both endpoints corrupt and so the reports are arbitrary. Previously, the set S were good in any configuration of identities consistent with the reports and the budget. But now, the bad party's budget increases by at least $g-1 \geq |S|$, and any configuration with the set S's identity changed to all bad is also consistent with the reports.

Therefore, no node is good in all configurations, and so no node can be detected as good. This strategy requires $m(G,g)+g-1$ nodes and prevents identification of one good node. Since $m(G)$ is the minimal number of bad nodes so that it is impossible to detect one good node, $m(G) \leq m(G,g)+g-1$. □

Now we continue to prove the $m(G', \delta|V'|+1) \geq m(G)$ direction of Claim 4.5.9. Assume towards contradiction that there exists a strategy that controls at least one node in the h-clique, prevents identification of $h+1$ good nodes, and requires fewer than $m(G)$ bad nodes in total. Suppose this strategy assigns a nodes in the h-clique as bad, where $1 < a < m(G) \leq |V|/2 \leq h/2$. Then $h-a > h/2 > m(G) > b$. Therefore, the rest of the h-clique forms a connected component with only good reports, and is of size $h-a$, which is larger than the bad party's budget $b < m(G)$, thus are declared as good. As a result, the bad party must prevent identification of $a+1$ good nodes in G with budget strictly less than $m(G)-a$. This contradicts the fact that $m(G)-a \leq m(G,a)-1 < m(G,a+1)$ by Claim 4.5.10.

Therefore, the strategy of controlling $m(G)$ nodes on G and let the h-clique be detected as good is an optimal strategy, $m(G', \delta|V'|+1) = m(G)$. □

Now, with Claim 4.5.9, we conclude that for any $1/2 \leq \delta < 1$, approximating $m(G, \delta|V|)$ within any constant must be SSE-hard. If not, we will obtain an efficient algorithm for approximating $m(G)$ by constructing a graph G' by adding a $\frac{\delta}{1-\delta}|V|$ clique to any graph G, for some δ, and approximate $m(G)$ by approximating $m(G', \delta|V'|+1)$, which is just $m(G', \delta'|V'|)$ for some other $0 < \delta' < 1$.

Theorem 4.5.5 implies a similar corollary about the SSE-hardness of seeding the nodes on a graph G given any constant multiple of the critical number $m(G, \delta|V|)$ to prevent detection of any arbitrary fraction of good nodes.

Corollary 4.5.11. *Assume the SSE Hypothesis and P ≠ NP. Fix any $\beta > 1$, and $0 < \delta < 1$. There does not exist a polynomial-time algorithm that takes as input an arbitrary graph $G = (V,E)$*

and outputs a set of nodes S with size $|S| \leq O(\beta \cdot m(G, \delta|V|))$, such that corrupting S prevents the central agency from finding $\delta|V|$ truthful nodes.

4.6 Acknowledgements

We would like to thank Paxton Turner and Vishesh Jain for useful conversations about corruption detection over the past year. We would like to thank Pasin Manurangsi for pointing us to the Austrin-Pitassi-Wu inapproximability result for treewidth.

4.7 Omitted Results

We give an NP-hardness result for computing $\min_k S_G(k) + k$ exactly. Note that this is insufficient to say anything about corruption detection, as $\min_k S_G(k) + k$ only gives a 2-approximation to the critical number $m(G)$, but we include this observation here as it may be of independent interest.

Theorem 4.7.1. *It is NP hard to compute* $\min_k S_G(k) + k$ *exactly.*

Proof. It is known that finding k-vertex separator for a graph is NP hard [105]. We present a reduction of the problem of computing $\min_k S_G(k) + k$ to the k-vertex separator problem.

Assume towards contradiction that there is a polynomial-time algorithm $\mathcal{A}$ for finding $\min_k S_G(k) + k$. Then for any graph G and any $M < |V|$, the minimal M-vertex separator of the graph $G = (V, E)$ can be found in the following way. Construct a graph $G' = (V', E')$, where

$$G' = G \cup \{n^2 \text{ disjoint M-cliques}\},$$

with $n \gg N := |V|$. Construct a second auxiliary graph $G'' = (V'', E'')$, such that

$$G'' = G' \cup \{kn + N \text{ disjoint } (n-1)\text{-cliques appended to each vertex of V'}\}.$$

Each $(n-1)$-clique is appended to a vertex of G' in the sense that each node of the clique is connected to the vertex in G' with an edge. The idea is to make each vertex in G' "n times larger".

Run the polynomial-time algorithm $\mathcal{A}$ for finding $\min_k S_{G''}(k) + k$ on graph G''. The algorithm outputs a vertex set $S'' \subseteq V''$, which divides G'' into connected components of with maximal size k''.

Lemma 4.7.2. *Let G'' be as constructed above, k'' and S'' be the output given by an algorithm that computes $\min_k S_{G''}(k) + k$. Then $k'' = nM$, and without loss of generality, the subset S'' contains only vertices from the original graph G. In other words, finding $\min_k S(k) + k$ of G'' is equivalent to finding the M-vertex separator of G. i.e.,*

$$\arg\min_k S_{G''}(k) + k = nM,$$

$$\min_k S_{G''}(k) + k = S_G(M) + nM.$$

Proof of Lemma 4.7.2. Let $f_{G''}(k) := S_{G''}(k) + k$, and let $f^*_{G''} := \min_k f_{G''}(k)$. Note there exists following upper bound for $f^*_{G''}$.

$$f^*_{G''} \le S_G(M) + nM$$

This is achieved by removing the M-vertex separator of G from G'' and divide $G''_{V''\setminus S_G(M)}$ into connected components with size at most nM.

Now we prove that $f^*_{G''}$ has to be exactly $S_G(M) + nM$ by showing that $f_{G''}(k) > f^*_{G''}$ for $k > nM$, and for $k < nM$.

1. $f_{G''}(k) > f^*_{G''}$ for all $k < nM$.

   For $k < nM$:
$$f_{G''}(k) \ge n^2 + k > S_G(M) + nM,$$

   because the separator has to include at least one vertex from each of the n^2 disjoint

nM-cliques in G''. This value $f_{G''}(k)$ is clearly larger than $S_G(M) + nM$ when $n \gg N > M$.

2. $f_{G''}(k) > f^*_{G''}$ for all $k > nM$.

   Claim 4.7.3. *We claim that it suffices to only consider k in the form of $k = nM + n\alpha$, where $\alpha \in \mathbb{Z}_+$. i.e. for any $k > nM$, $f_{G''}(k) \geq f_{G''}(nM + n\alpha)$ for some $\alpha \in \mathbb{Z}_+$.*

   Proof of Claim 4.7.3. Call the nodes in G to which each of the n-clique is appended to (while constructing G'') the **center** of the n-clique in G''. If k cannot be expressed in the form of $nM + n\alpha$, this means the corresponding separator S contain some non-center nodes of the n-cliques in G''.

   If the center $\notin S$, while some other node(s) of the clique $\in S$, there exists another S^*, $|S^*| < |S|$ that includes the center instead of the other node(s), and suffice to be a k-vertex separator. This is because after the removal of the center node, the rest of the clique can be of size at most $(n-1)$, and $k > nM > n - 1$.

   Suppose the center $\in S$, while some of the other node(s) of the clique also $\in S$, in order to obtain a k-vertex separator. Then S^* that only contains center will suffice to be k-vertex separator, because $k > n$. □

   By Claim 4.7.3, for any $k > nM$, $f_{G''}(k) \geq f_{G''}(nM + n\alpha)$ for some $\alpha \in \mathbb{Z}_+$. In words, there is never any incentive to include any non-center nodes of an n-cliques in separator S. Without loss of generality, $S \subseteq V_G$, and $k = nM + n\alpha \geq nM + n$.

   $$f_{G''}(nM + n\alpha) > nM + n > S_G(M) + nM$$

   when $n \gg N$.

Summarizing 1 and 2, we conclude that

$$f^*_{G''} = f_{G''}(nM) = S_G(M) + nM$$

□

This gives us a polynomial algorithm to find any M-vertex separator for any graph G, and any value M. This contradicts the fact that computing M-vertex separator is NP-hard. Therefore, there does not exist polynomial time algorithm for computing $\min S_G(k)+k$. □

Chapter 5

Opinion Polarization

5.1 Introduction

We conclude this thesis by discussing a model for opinion polarization. Opinion polarization is a widely acknowledged social phenomenon, especially in the context of political opinions [48, 152, 83], leading to recent concerns over "echo chambers" created by mass media [136] and social networks [35, 129, 12, 11, 52]. The objective of this paper is to propose a simple, geometric model of the dynamics of polarization where the opinion structure (that is, correlations within the population's opinions on various topics) can change under influence of advertising or political campaigns. Many models have been proposed to explain how polarization arises, and this remains an active area of research [127, 10, 126, 77, 110, 15, 38, 41, 98, 132, 140].

Unlike the previous chapters on inference which modified existing mathematical models, in this chapter we create a new mathematical model for opinion polarization. Our attempt aims at simplicity over complexity. As opposed to a large majority of previous works addressing opinion polarization, we neglect the social network structure and interactions between individuals. Instead, we focus on influences of advertising or political *campaigns* that reach a wide segment of the population. Our main behavioral assumption is *biased assimilation* [108]: people tend to be receptive to opinions they agree with, and antagonistic to opinions they disagree with.

Another distinguishing feature of our model is the *multi-dimensional* setting, reflecting the fact that campaigns can touch on many diverse topics. For example, in the context of American politics, one might wonder why there exists a significant correlation between opinions of individuals on, say, abortion access, gun rights and urgency of climate change [33]. Our model attempts to illustrate how such correlations between opinions can arise as a (possibly unintended) effect of advertising exploiting different topics and social values.

Our model falls into framework of inference that is similar to that of Chapter 4. We have two types of parties: influencers, who want to push a campaign, and individuals, who have a host of opinions about different topics. Given the opinions of different individuals, the goal of the influencers is to bring as many people as possible as close to their own campaign values as possible. For example, if the influencer were selling a certain car, their goal is to make as many people as possible have a positive view of this car. The influencers know how their advertisements affect opinions in the population, and want to solve an inference problem to determine what is the best sequence of advertisements to achieve their goal.

In mathematical terms, we consider a population of agents with preexisting opinions represented by vectors in $\mathbb{R}^d$, normalized such that the Euclidean length of each vector is 1. Each coordinate represents a distinct topic, and the value of the coordinate reflects the agent's opinion on the topic, which can be positive or negative. We then consider a sequence of *interventions* affecting the opinions. An intervention is also a unit vector in $\mathbb{R}^d$, representing the set of opinions expressed in, e.g., an advertising campaign or "news cycle". Therefore, all opinions and interventions in our model lie on the unit sphere in $\mathbb{R}^d$.

We model the effect of intervention v on an agent's opinion u in the following way. Supposing an agent starts with opinion $u \in \mathbb{R}^d$, after receiving an intervention v they will update the opinion to the unit vector proportional to

$$w = u + \eta \cdot \langle u, v \rangle \cdot v \,, \tag{5.1}$$

where $\eta > 0$ is a global parameter that controls the influence of an intervention. Most of our results do not depend on a choice of η and in our examples we often take $\eta = 1$ for the sake of simplicity. Smaller values of η could model campaigns with limited persuasive power.

Intuitively, the agent evaluates the received message in context of their existing opinion, and assimilates this message weighted by their "agreement" with it. Our model exhibits biased assimilation in that if the intervening opinion v is positively correlated with an agent's opinion u, then after the update the agent opinion moves towards v, and conversely, if v is negatively correlated with u, then the update moves u away from v and towards the opposite opinion $-v$.

There are multiple scenarios that our model could reflect. One way to think of the intervention is as an exposure to persuasion by a political actor, like a political campaign message. Another example, in the context of marketing, is a product advertisement that exploits values besides the quality of the product. In that context, we can think of one of the d coordinates of the opinion vector as representing one's opinion on a product being introduced into the market and the remaining coordinates as representing preexisting opinions on other (e.g., social or political) issues. Then, an intervention would be an advertising effort to connect the product with a certain set of opinions or values [163]. Some examples are corporate advertising campaigns supporting LGBT rights [153] or gun manufacturers associating their products with patriotism and conservative values [141]. More broadly, another example of an intervention could be a company (e.g., a bank or an airline [51]) announcing its refusal to do business with the gun advocacy group NRA. It seems plausible to us that such advertising strategies can have a double effect of convincing potential customers who share relevant values and antagonizing those who do not.

Furthermore, it seems conceivable (and, as shown later, will provably happen in some settings in our model) that such interventions, even if intending mainly to increase sales and without direct intention to polarize, can have a side effect of increasing the extent of polarization in the society. For example, it might be that, in a population with initial opinions distributed uniformly, a number of interventions introduces some weak correlations. In our model, these correlations can be profitably exploited by advertisers in subsequent interventions. As a side effect, the interventions strengthen the correlations and increase polarization.

For example, suppose that after various advertising campaigns, people who tend to like item A (say, electric cars) tend to be liberal, and people who like a seemingly unrelated item B (say, firearms) tend to be conservative. This may result from the advertisers exploiting some obvious connections, e.g., between electric cars and responding to climate change, and between firearms and respect for the military. Subsequently, future advertising efforts for electric cars may feature other values associated with liberals in America to appeal to potential consumers: an advertisement might show a gay couple driving to their wedding in an electric car. Similarly, future advertisements for firearms may appeal to conservative values for similar reasons. The end result can be that the whole society becomes more polarized by the incorporation of political topics into advertisements.

Throughout the paper, we analyze properties of our model in a couple of scenarios. With respect to the interventions, we consider two scenarios: either there is one entity (an *influencer*) trying to persuade agents to adopt their opinion or there are two competing influencers pushing different agendas. With respect to the time scale of intervations, we also consider two cases: the influencer(s) can apply arbitrarily many interventions, i.e., the *asymptotic* setting, or they need to maximize influence with a limited number of interventions, i.e., the *short-term* setting. The questions asked are: (i) What sequence of interventions should be applied to achieve the influencer's objective? (ii) What are the computational resources needed to compute this optimal sequence? (iii) What are the effects of applying the interventions on the population's opinion structure? We give partial answers to those questions. The gist of them is that in most cases, applying desired interventions increases the polarization of agents.

5.1.1 Outline and main results

In the asymptotic setting with one influencer with a desired campaign agenda, we show that the optimal campaigning strategy does not necessarily push the campaign agenda directly at every step. Instead, it finds a hemisphere with the largest number of initial opinions, concentrates the opinions in this hemisphere around an arbitrary point, and only in the last stage nudges them gradually towards the target agenda (Theorem 5.4.3). We then show that it is computationally hard to approximate this densest hemisphere (and therefore the optimal strategy) to any additive factor (Theorem 5.4.11). Notably, a very strong notion of polarization emerges from our dynamic: there exists a pair of antipodal points such that all opinions converge to one of them. Of course we are not suggesting that this should occur in practice. Rather, we expect more realistic dynamics to be more complicated, with similar, but weaker forms of polarization occurring on a shorter time scale.

In the asymptotic setting with two competing influencers, each pushing their respective campaign agenda with each intervention, we show that all opinions will converge to the convex cone between the two campaign agendas (Theorem 5.4.15). One might hope that having multiple advertisers can make the resulting opinions more spread-out, but in fact we prove that the same strong form of polarization emerges if the correlation between the campaign agendas is high enough: the opinions of the population concentrate around two antipodes moving around in the convex cone of the two agendas (Theorem 5.4.16).

In the short-term setting with one influencer, a similar result for the optimal campaign holds: the optimal campaign is equivalent to finding a spherical cap containing the largest number of initial opinions (Theorem 5.8.1). Furthermore, we describe a simple case study with two agents illustrating polarization as an externality imposed on the society by the interventions. We define the notion of polarization cost and trace the correlation between the agents for various interventions, showing that in some settings it might be desirable for the influencer to apply an intervention that increases the extent of disagreement between the agents (Section 5.4.3).

We additionally show that in two dimensions, starting from opinions uniformly distributed on a sphere, random interventions also lead to the strong form of polarization to antipodes (Theorem 5.4.1).

In Section 5.5 we present two examples with initial opinions distributed uniformly at random. We illustrate how the opinions evolve and polarization increases as a result of interventions. The first example shows a clear pattern of polarization when a fixed intervention is applied repeatedly, in the strong sense that opinions tend to two antipodes. In the second example we apply two orthogonal interventions in an alternating fashion. Interestingly, the opinions do not completely polarize, suggesting that polarization in our model is not inevitable, but biased assimilation in multiple dimensions does tend to lead to degenerate opinion structures (antipodes as a 1-dimensional and ring as a 2-dimensional degenerate structure). The results of the simulations are illustrated in Figures 5-3 and 5-4.

5.1.2 Limitations

Before we describe our results, we discuss some limitations of our approach. We present a basic model intended to capture one mode of emergence of polarization. Most importantly and in contrast to majority of existing literature, our model neglects opinion changes induced by interactions between individuals. Furthermore, we do not address aspects such as replacement of the population or unequal exposure and effects of the interventions. We do not consider any external influences on the population in addition to the interventions. We also do not confront theoretical and empirical research suggesting that in certain settings exposure to conflicting views can decrease polarization [133, 118, 54, 53] or even questioning the overall extent of polarization in the society [49, 16]. As a matter of fact, applying a sequence of random interventions in our model results in the polarization of opinions (but we also present an example where total polarization does not occur). We leave addressing these limitations for the future.

While we sometimes discuss the uniform distribution of initial opinions on $\mathbb{R}^d$, we do not claim that it is the most plausible one and we do not make assumptions about the initial distribution in most of our results. We assume that any group of topics can be combined into an intervention with the effect given by (5.1). We expect that a more plausible model might feature some "internal" correlations between topics in addition to "external" correlations arising out of the agents' opinion structure. For example, topics may have related meaning, causing inherent correlations between corresponding opinions (e.g., being positive on renewable energy and recycling). Furthermore, there are certain topics (e.g., undesirability of murder) on which nearly all members of the population share the same inclination. As a matter of fact, it is common for marketing strategies to exploit unobjectionable social values (see, e.g., [163]). However, we presume that under suitable circumstances (e.g., due to inherent correlations we just mentioned) the "polarizing" topics might present a more appealing alternative for a campaign. Our model concerns such a case, where the "unifying" topics might be neglected and excluded from the analysis. We note that other works have also suggested that focusing on polarizing topics may be appealing for campaigns (see, e.g., [132]).

One property of our model is that an effect of an intervention using opinion v is exactly the same as for the opposite opinion $-v$. This might look like a cynical assumption about human nature, but arguably it is not entirely inaccurate. For example, experiments on social media show that not only exposure to similar ideas increases polarization (the "echo chamber" effect), but also exposure to the opinions opposite to one's own causes beliefs to become more extreme and polarized [11]. Furthermore, in our model this effect occurs only if all the components of an opinion are negated.

We also note that our representation of opinions contains some ambiguities. A "weak" opinion $u_{i,k} \approx 0$ might signify each of: neutrality, lack of confidence, or lack of interest in a given subject. More generally, we assume that the agents have a "fixed budget" (one unit in Euclidean norm) of opinions that they always fully use. On the one hand, one might expect that different kinds of opinions will update in different manners: a confident neutral opinion might be harder, while a weak extreme opinion easier to change. On the other hand, there are psychological reasons to expect that, e.g., "issue interest" and "extremity of opinion" are correlated [104, 14] (see also the discussion in [15]).

Finally, we do not study how small modifications of our model, e.g., using a different norm or a different normalization method, modify its behavior.

5.2 Related Works

As mentioned, there is a multitude of modeling and empirical works studying opinion polarization in different contexts [127, 10, 126, 77, 110, 120, 15, 38, 41, 98, 140, 13, 132, 11]. Broadly speaking, previous works have proposed various possible sources for polarization, including peer interactions, bias in individuals' perceptions, and global information outlets.

There is an extensive line of models of opinion exchange on networks with peer interactions, where individuals encounter neighboring individuals' opinions and update their own opinions based on, e.g., pre-defined friend/hostile relations [151], or the similarity and relative strength of opinions [118], etc. This branch of work often attributes polarization to homophily of one's social network [38] that is induced by the self-selective nature of social relations and segregation of like-minded people [165] and exacerbated by the echo chamber effect of social media [129].

A parallel proposed mechanism are psychological biases in individuals' opinion formation processes. One example is biased assimilation [108, 38, 15, 11]: the tendency to reinforce one's original opinions regardless if other encountered opinions align with them or not. For example, [11] observed that even when social media users are assigned to follow accounts that share opposing opinions, they still tend to hold their old political opinions and often to a more extreme degree. On the modeling side, [38] showed that DeGroot opinion dynamics with the biased assimilation property on a homophilous network may lead to polarization.

Existing works have also proposed models where polarization happens even when information is shared globally [168, 120]. For example, [120] propose a model where competition for readership between global information outlets causes news to become polarized in a single-dimensional setting. Another example is [168], a classical work on the formation of mass opinion. It theorizes that each individual has political dispositions formed in their own life experience, education and previous encounters that intermediate between the message they encounter and the political statement they make. Therefore, hearing the same political message can cause different thinking processes and changes in political preferences in different individuals.

It is noteworthy that the majority of previous work focuses on polarization on a single topic dimension. Two exceptions are [15], which studies biased assimilation with opinions on multiple topics and [16] that observed non-trivial correlation between people's attitudes on different issues. As a matter of fact, [15] uses a different updating rule to observe dynamics that differ from our work: in their simulations, polarization on one issue typically does not result in polarization on others. There is also a class of models [10, 126, 110] that concern multi-dimensional opinions where an opinion on a given topic takes one of finitely many values (e.g., + or -). These models do not seem to have a geometric structure of opinion space similar to ours and usually focus on formation of discrete groups in the society rather than total polarization. Another model in [130] uses a geometric (affine) rule of updating multi-dimensional opinions. Unlike us, they seem to be modeling pre-existing, "intrinsic" correlations between topics rather than the emergence of new ones and they are concerned mostly with convergence and stability of their dynamics.

A related paper [132] contains a geometric model of opinion (preference) structures. Both this and our model propose mechanisms through which information outlets acting for their own benefit can lead to increased disagreement in the society. The key difference between [132] and our model is that, in [132], the population's preferences are static and do not update, but the firms are free to choose how they acquire information. By contrast, in our model, the influencers have pre-determined ideologies and compete to align agents' opinions with their own. In other words, [132] focuses on modeling of competitive information acquisition, and our paper on modeling the influence of marketing on the public opinion.

Our model suggests that under the conditions of biased assimilation, opinion manipulation by one or several global information outlets can unintentionally lead to a strong form of polarization in multi-dimensional opinion space. Not only do people polarize on individual issues, but also their opinions on previously unrelated issues become correlated. This form of polarization is particularly related to *issue alignment* [16] discussed in political science and sociology literature. Issue alignment refers to an opinion structure where the population's opinions on multiple (relatively independent) issues correlate. It is related to *issue radicalization*, where the opinions polarize for each issue separately. Compared to issue radicalization, issue alignment is theorized to pose more constraints on the opinions an individual can take, resulting in polarized and clustered mass opinions even when the public opinions are not extreme in any single topic, and presenting more obstacles for social integration and political stability [16]. In light of this, one way to view our model is as a mathematical mechanism by which this strong form of polarization can arise and worsen due to companies', politicians', and the media's natural attempts to gain support from the public.

On the more technical side, we note that our update equation bears similarity to Kuramoto model [84] for synchronization of oscillators on a network in the control literature. In this model, each oscillator i is associated with the point θ_i on the two-dimensional sphere, and i updates its point continuously as a function of its neighbors' points θ_j:

$$\dot{\theta}_i = \omega_i + \frac{K}{N}\sin(\theta_j - \theta_i),$$

where K is the *coupling strength* and N is the number of nodes of the network. In two dimensions, our model can be compared to Kuramoto model with $\omega_i = 0$ on a star graph, with the influencers at the center of the star connected to the entire population, where the influencers' opinions do not change and the update strength is qualitatively similar to $\sin((\theta_v - \theta_u)/2)$ (see Equation 5.26 in Section 5.9.2 for the actual function). However, we note a crucial difference: in the Kuramoto dynamic, θ_i always moves towards θ_j, i.e. nodes always move towards synchronization, but in our dynamic, opinions θ_i are allowed to move further away from θ_j when the angle between their opinions are obtuse. In addition, the central node in our model can be strategic in choosing their positions, while the central node in Kuramoto model follows the synchronization dynamics of the system. We think this property provides a better model for opinion interactions.

5.3 The Model

We consider a group of n *agents*, whose opinions are represented by unit vectors in $\mathbb{R}^d$, where we think of each component as representing a distinct topic. It might be useful to think of n as much larger than d, though our results do not assume this. We will look into how those opinions change after receiving a sequence of *interventions*. Each intervention is also a unit vector in $\mathbb{R}^d$, representing the opinion contained in a message that the influencer (e.g., an advertiser) broadcast to the agents. Our model features one parameter: $\eta > 0$, signifying how strongly an intervention influences the opinions.

After each intervention, the agents update their opinions by moving towards or away from the intervention vector, depending on whether or not they agree with it (which is determined by the inner product between the vector v and the opinion vector), and normalizing suitably. Suppose the agents' initial opinions are $u_1, \ldots, u_n$, $\|u_i\| = 1$, and an intervention v is applied, then the updated opinions $u_1', \ldots, u_n'$ are given by

$$w_i = u_i + \eta \langle u_i, v \rangle \cdot v \ , \tag{5.2}$$

$$u_i' = \frac{w_i}{\|w_i\|} \ , \tag{5.3}$$

where we note that

$$\|w_i\|^2 = \langle w_i, w_i \rangle = 1 + (2\eta + \eta^2) \langle u_i, v \rangle^2 \tag{5.4}$$

by expanding out the definition of w_i. In particular, this implies that $\|w_i\| \geq 1$, and consequently that u_i' is well-defined. The norm in (5.3) and everywhere else throughout is the standard Euclidean norm. Note that applying v and $-v$ always yields the same results.

Objectives We analyze the strategy of influencers in several settings.

In an **"asymptotic scenario"**, the influencer wants to apply an infinite sequence of interventions $v^{(1)}, v^{(2)}, \ldots$, that maximizes how many out of the n agent opinions converge to the target vector v. As is standard, we say that a sequence of vectors $u^{(1)}, \cdots, u^{(t)}, \ldots$ converges to a vector v if $\lim_{t\to\infty} \|u_i^{(t)} - v\| = 0$. One way to interpret this scenario is that a campaigner wants to establish a solid base of support for their party platform.

In a **"multiple-influencer scenario"**, two influencers (such as two companies or two parties) who have different objectives apply their two respective interventions on the population in a certain order. We ask how the opinions change under such competing influences. This scenario can be interpreted as two parties campaigning their agendas to the population.

In a **"short-term scenario"**, the influencer is assumed to be an advertiser, and the opinions of the product are expressed in the last coordinate of opinion vectors $u_{i,d}$. The influencer assumes some fixed threshold $0 < T < 1$ and an upper bound K on the number of interventions, and asks, given n opinions $u_1, \ldots, u_n$, how to choose $v^{(1)}, \cdots, v^{(K)}$ in order to maximize the number of time-K opinions $u_1^{(K)}, \ldots, u_n^{(K)}$ with $u_{i,d}^{(K)} > T$. One interpretation is that advertisers only have a limited number of opportunities to publicize their products to consumers, and consumers with $u_i^{(K)} > T$ will decide to buy the product after the interventions $v^{(1)}, \cdots, v^{(K)}$ are applied.

There are several possible variants of our model that we find potentially interesting but do not address in this paper. For example:

- "Targeting", where the influencer can select subgroups of the population and apply interventions groupwise.

- Perturbing preferences with noise after each step.

- Replacement of the population, e.g., introducing new agents with "fresh" opinions or removing agents that stayed in the population for a long time or who already "bought" the product, i.e., exceeded the threshold $u_{i,d} > T$. For example, this could correspond to "one-time" purchase product like a house or a fridge, or situations where the customer's opinion is more difficult to change as time passes.
- Models where the initial opinions are not observable or partially observable.
- Expanding the model by adding peer effects and social network structure and exploring the resulting dynamics of polarization and opinion formation.
- Strategic competing influencers: in the studied scenarios with competing influencers, we assume that they apply fixed interventions. One can ask: supposing the influencers have their own target opinions, what is each campaigner's optimal sequence of messages in face of the other campaigner? Then, resulting equilibrium of opinion formation could be analyzed.

5.4 Outline and Summary of Main Results

In this section, we present our results in different settings, with most of the proofs deferred to later sections.

5.4.1 Asymptotic scenario: random interventions polarize opinions

We analyze the long-term behavior of our model in a simple random setting. We assume that, in dimension $d = 2$, at the initial time $t = 1$ we are given n opinion vectors $u_1^{(1)}, \ldots, u_n^{(1)}$. Subsequently, we apply a sequence of interventions $v^{(1)}, v^{(2)}, \ldots$, such that each intervention $v^{(t)}$ is sampled i.i.d from the uniform distribution on the unit circle S^1. More precisely, at time t we apply the random intervention $v^{(t)}$ to every opinion vector $u_i^{(t)}$, obtaining a new opinion $u_i^{(t+1)}$.

We want to show that the opinions almost surely polarize as time t goes to infinity. We need to be careful about defining the notion of polarization: since the interventions change at every time step, the opinions cannot converge to a fixed vector. Instead, we show that for every pair of opinions the angle between them converges either to 0 or to π. More formally:

Theorem 5.4.1. *For any fixed opinions $u_1^{(1)}$ and $u_2^{(1)}$ and a sequence of uniform i.i.d. interventions, we have*

$$\mathbb{P}\left[\|u_1^{(t)} - u_2^{(t)}\| \to 0 \vee \|u_1^{(t)} + u_2^{(t)}\| \to 0\right] = 1 \ .$$

This leads to the following corollary for any finite number of agents:

Corollary 5.4.2. *For any fixed opinions $u_1^{(1)}, \dots, u_n^{(1)}$ and a sequence of uniform i.i.d. interventions, almost surely, there exists $S \subseteq \{1, \dots, n\}$ such that the diameter of the set*

$$\left\{(-1)^{\mathbb{1}[i \in S]} \cdot u_i^{(t)} : i \in \{1, \dots, n\}\right\}$$

converges to zero.

Theorem 5.4.1 is proved in Section 5.6 using martingale convergence. We believe the theorem holds also for $d \geq 3$, but our proof does not apply to this case.

5.4.2 Asymptotic scenario: finding densest hemishpere

In this section we study the asymptotic scenario with one influencer. In this setting, the influencer wishes to propagate a campaign agenda $v^* \in \mathbb{R}^d$. We assume that the influencer can use an unlimited number of interventions and their objective is to make the opinions of as many agents as possible to converge to v^*. More precisely, given the preexisting opinions of n agents, $u_1, \dots, u_n$, we want to find a sequence of interventions, $v^{(1)}, v^{(2)}, v^{(3)} \dots$ that maximizes the number of agents whose opinions converge to v^*.

The thrust of our results is that finding a good strategy for the influencer is computationally hard. However, both the optimal strategy and some natural heuristics result in the polarization of agents.

We first argue that the problem of finding an optimal strategy is equivalent to identifying an open hemisphere that contains the maximum number of agents. An *open hemishpere* is an intersection of the unit sphere with a *homogeneous open halfspace* of the form $\left\{u \in \mathbb{R}^d : \langle u, v\rangle > 0\right\}$ for some $v \in \mathbb{R}^d$.

Theorem 5.4.3. *For any v^*, there exists a strategy to make at least k agents converge to v^* if and only if there exists an open hemisphere containing at least k of the opinions $u_1, \ldots, u_n$.*

Proof of Theorem 5.4.3. First, we prove that the hemisphere condition is sufficient for the existence of a strategy to make the agents' opinions converge (Claim 5.4.4). Then we prove the trickier direction: that the hemisphere condition is also *necessary* for the existence of such a strategy (Claim 5.4.9).

Claim 5.4.4. *If opinions $u_1, \ldots, u_k$ are contained in an open hemisphere, then there is a sequence of interventions making all of $u_1, \ldots, u_k$ converge to v^*.*

Proof. By definition of open hemisphere, there is a vector $a \in \mathbb{R}^d$ such that $\langle a, u_i\rangle > 0$ for every agent $i = 1, \ldots, k$. By (5.2), it is clear that repeated application of a makes all the points converge to a as time $t \to \infty$.

After all the points are clustered close enough to a, by a similar argument they can be "moved around" together towards another arbitrary point v^*. For example, if $\langle v^*, a\rangle > 0$, the intervention v^* can be applied repeatedly. If $\langle v^*, a\rangle \leq 0$, one can proceed in two stages: First applying an intervention proportional to $(v^* + a)/2$, and then v^*. □

Remark 5.4.5. *As a possible interpretation of the mechanism in Claim 5.4.4, it is not unheard of in campaigns on political issues to use an analogous strategy. First, build a consensus around a (presumably compromise) opinion. Then, "nudge" it little by little towards another direction.*

In an extreme case one can imagine this mechanism even flipping the opinions of two polarized clusters. One example of this could be reversal of the opinions on certain issues of 20th *century Republican and Democratic parties in the US (this particular phenomenon can be found in many texts, e.g. [100]).*

To prove the other direction of Theorem 5.4.3, we will rely on the notions of conical combination and convex cone. A *conical combination* of points $u_1, \ldots, u_n \in \mathbb{R}^d$ is any point of the form $\sum_{i=1}^n \alpha_i u_i$ where $\alpha_i \geq 0$ for every i. A *convex cone* is a subset of $\mathbb{R}^d$ that is closed under finite conical combinations of its elements. Given a finite set of points $S \subseteq \mathbb{R}^d$, the convex cone *generated* by S is the smallest convex cone that contains S.

Claim 5.4.6. *Let $\varepsilon > 0$. Suppose that for a given sequence of interventions the opinions $u_1, \ldots, u_n$ get within ℓ_2-distance ε to some point v^*. Then, for any unit vector u_{n+1} that lies in the convex cone of $u_1, \ldots, u_n$ intersected with the unit sphere, we have that u_{n+1} gets within distance at most 2ε of v^*.*

Proof. First, we prove that if u_{n+1} lies in the convex cone of $u_1, \ldots, u_n$, then after applying one intervention v the new opinion u'_{n+1} lies in the convex cone of $u'_1, \ldots, u'_n$. This shows that points in the convex cone always stay within the convex cone, by induction.

To prove this, we can simply write out u'_{n+1}, using the relation $u_{n+1} = \sum_{i=1}^n \lambda_i u_i$ (where we use the notation $u \propto v$ to mean that $u = c \cdot v$ for some constant $c > 0$):

$$\begin{aligned}
u'_{n+1} &\propto u_{n+1} + \eta \langle u_{n+1}, v \rangle \cdot v \\
&= \sum_{i=1}^n \lambda_i u_i + \eta \cdot \sum_{i=1}^n \lambda_i \langle u_i, v \rangle \cdot v \\
&= \sum_{i=1}^n \lambda_i (u_i + \eta \cdot \langle u_i, v \rangle \cdot v) \\
&= \sum_{i=1}^n \lambda_i \cdot c_i u'_i \qquad (5.5)
\end{aligned}$$

where the constants in (5.5) are $c_i := \left\| u_i + \eta \cdot \langle u_i, v \rangle \cdot v \right\|$. Specifically, they are all nonnegative.

Next, we show that if $u_1, \ldots, u_n$ are within distance ε of v^*, then so any point in their convex cone intersected with the unit sphere is distance at most 2ε away. Let $u := \sum_{i=1}^n \lambda_i u_i$ be such a point. Let u^* denote u with normalized weights; that is, $u^* = \sum_{i=1}^n \eta_i u_i$ where $\eta_i := \lambda_i / \sum_{j=1}^n \lambda_j$.

$$\|u - v\| \leq \|u - u^*\| + \|u^* - v\|$$

Note that u is the closest point to u^* on the unit sphere; indeed for any point z on the unit sphere

$$\|u^* - z\|_2^2 = \|u^*\|_2^2 + 1 - 2\langle u^*, z\rangle \geq \|u^*\|_2^2 + 1 - 2\|u^*\|_2$$

by Cauchy-Schwarz. Furthermore, $\|u^* - v\| \leq \varepsilon$, which follows from the triangle inequality. Hence, we get that $\|u - v\| \leq 2\varepsilon$ as desired. □

Claim 5.4.7. *Suppose there are two opinions u_1, u_2 that are antipodal, i.e., $u_1 = -u_2$. Then these two opinions will remain antipodal in future time steps. In particular, they will never converge to a single point.*

Proof. This follows directly from (5.2), noting that, for any intervention v, we have $u_1 + \eta \cdot \langle u_1, v\rangle \cdot v = -(u_2 + \eta \cdot \langle u_2, v\rangle \cdot v)$. □

We will also use the following consequence of the separating hyperplane theorem:

Fact 5.4.8. *A collection of unit vectors $a_1, \ldots, a_n$ cannot be placed in an open hemisphere if and only if the zero vector lies in the convex hull of $a_1, \ldots, a_n$.*

Now we are ready to establish the reverse implication in Theorem 5.4.3.

Claim 5.4.9. *Suppose that we start with agent opinions $u_1, \ldots, u_n$ and that there is no hemisphere that contains M of those opinions. Then, there is no strategy that makes M of the opinions converge to the same point.*

Proof. Assume towards contradiction that there exists a strategy that makes M opinions converge to the same point, and assume wlog that they are $u_1, \ldots, u_M$. By assumption, we know that there is no hemisphere that contains all of $u_1, \ldots, u_M$, hence, by Fact 5.4.8, there is a convex combination of $u_1, \ldots, u_M$ that equals 0. Therefore, there is also a conical combination of $u_1, \ldots, u_{M-1}$ that equals $-u_M$, where wlog we assume that the coefficient on u_M is initially nonzero. By Claim 5.4.6, we conclude that if $u_1, \ldots, u_{M-1}$ converge to the same point, then so does $-u_M$. But that means that $-u_M$ and u_M converge to the same point, which is a contradiction by Claim 5.4.7. □

□

Remark 5.4.10. *One consequence of Theorem 5.4.3 is that if the agent opinions are initially distributed uniformly on the unit sphere, an optimal strategy converging as many opinions as possible to v^* results, with high probability, in dividing the population into two groups of roughly equal size, where the opinions inside each group converge to one of two antipodal limit opinions (i.e., v^* and $-v^*$).*

As mentioned, Theorem 5.4.3 implies that an optimal strategy for the influencer is to compute the hemisphere that contains the most opinions and then apply the procedure from Claim 5.4.4 to converge the opinions from this hemisphere to v^*.

The densest hemisphere problem turns out to be equivalent to the previously studied problem of *learning noisy halfspaces*, allowing us to apply known algorithmic and computational hardness results. In particular, applying a work by Guruswami and Raghavendra [68] we show in Section 5.7 that it is computationally difficult to even approximate the densest hemisphere in a strong sense:

Theorem 5.4.11. *Unless P=NP, for any $\varepsilon > 0$, there is no polynomial time algorithm that distinguishes between instances of densest hemisphere problem such that, letting $D := \{u_1, \ldots, u_n\}$:*

- *Either there exists a hemisphere H such that $|D \cap H|/n > 1 - \varepsilon$.*
- *Or for every hemisphere H we have $|D \cap H|/n < 1/2 + \varepsilon$.*

Consequently, unless P=NP, there is no polynomial time algorithm that, given an instance D that has a hemisphere with density more than $1 - \varepsilon$, outputs a hemisphere with density more than $1/2 + \varepsilon$.

At the same time, [20] (relying on earlier work [19]) shows that there exists an algorithm that finds a dense hemisphere provided that this hemisphere is stable in the sense that it remains dense even after a small perturbation of its separating hyperplane:

Theorem 5.4.12 ([20]). *For every $\eta > 0$, there exists a polynomial time algorithm that, given an instance $D = \{u_1, \dots, u_n\}$ of the densest hemisphere problem, provides the following guarantee:*

If there exists a halfspace $H_\eta = \{x : \langle v, x\rangle > \eta\}$ such that $|D \cap H_\eta|/n > \alpha$, then the algorithm outputs a hemisphere corresponding to a homogeneous halfspace $H = \{x : \langle w, x\rangle > 0\}$ such that $|D \cap H|/n > \alpha$.

In other words, if there exists a hemisphere that contains many opinions, and the opinions do not lie close to the separating hyperplane, there is an efficient algorithm to find this hemisphere, which can then be used to persuade the agents.

5.4.3 Short-term scenario: polarization as externality

The analysis of asymptotic setting with unlimited interventions tells us what is feasible and what is not. A fundamentally different question is how to persuade as many as possible with limited number of interventions. This is motivated by bounded resources or time that usually allow only limited placements of campaigns and advertisements. Furthermore, arguably only the initial interventions can be considered effective: in the long run the opinions might shift due to external factors and become more unpredictable and harder to control. Therefore, in this section we discuss influencer strategies when it has only one intervention at its disposal, and its goal is to get as many agents as possible to have opinions close to v^*. Throughout this section, we fix $\eta = 1$ in Equation 5.1, so an opinion u is updated to be proportional to $w = u + \langle u, v\rangle \cdot v$.

We consider a simple example that features only two opinions and one influencer who is allowed one intervention. We imagine a new product being introduced into the market such that the agents are initially agnostic about it, i.e., $u_{i,d} = 0$ for $i = 1, 2$. Given an intervention v, we are interested in two issues: First, what will be new opinions of agents about the product $u'_{i,d}$? Second, assuming that the initial correlation between opinions is

$c = \langle u_1, u_2 \rangle$, what will be the new correlation $c' = \langle u_1', u_2' \rangle$? We think of the correlation as a measure of agreement between the agents and therefore interpret differences in correlation as changes in the extent of polarization.

To this end, we introduce notions of unifying and antagonizing interventions corresponding to two natural strategies:

Definition 5.4.13. *The* unifying intervention *is an intervention that maximizes* $\min(u_{1,d}', u_{2,d}')$. *The* antagonizing intervention *maximizes* $\max(u_{1,d}', u_{2,d}')$.

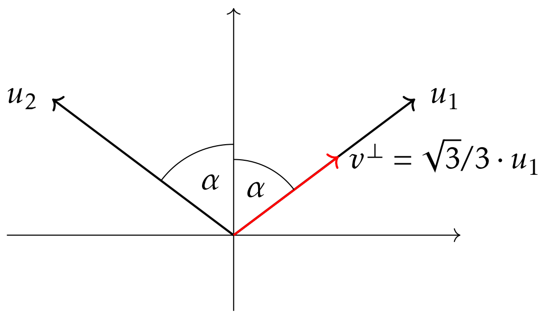
The motivation for this definition is as follows. If the opinions $u_{i,d}'$ correlate with the probability of the agent buying the product (e.g., an agent will consider buying only if their opinion exceeds certain threshold $u_{i,d}' > T$), and assuming that inducing as many sales as possible is the only thing that the influencer cares about, there are two natural choices for the intervention. One option is to apply v that will yield new opinions u_1', u_2' such that $\min(u_{1,d}', u_{2,d}')$ is maximized. This corresponds to the case when the influencer appeals to both agents with the hope of inducing two sales. The other case is to appeal only to one of the agents, for example, the first agent, disregarding the second agent and concentrating only on one possible sale.

Due to the geometrical nature of our model, to analyze unifying and antagonizing strategy in this setting, we can, without loss of generality, assume that $d = 3$ and that the initial opinions are given by

$$u_1 := (\sin\alpha, \cos\alpha, 0), \qquad u_2 := (-\sin\alpha, \cos\alpha, 0), \tag{5.6}$$

for $0 \le \alpha \le \pi/2$. Accordingly, we have $c = \cos^2\alpha - \sin^2\alpha = \cos(2\alpha)$. In particular, $\alpha = 0$ means that the agents are in full agreement, $\alpha = \pi/4$ corresponds to the case of orthogonal opinions and $\alpha = \pi/2$ is the case where the opinions are antipodal. The unifying and antagonizing interventions can be illustrated as shown in Figure 5-1 (see Section 5.8.1 for fuller derivation):

Rather intuitively, the unifying intervention has the form $v_{\mathrm{uni}} = (0, v_2, v_3)$, i.e., its



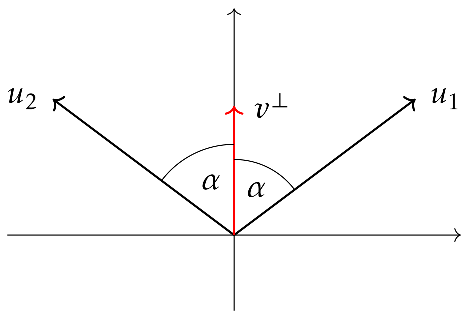


Figure 5-1: *The projection of antagonizing (left) and unifying (right) interventions onto the first two dimensions.*

projection onto span of u_1 and u_2 is a bisector of those two opinions. On the other hand, the projection of the antagonizing intervention v_{ant} is parallel to u_1 (or u_2). As suggested by the names, applying the antagonizing intervention results in the correlation c'_{ant} that is smaller (i.e., more polarized) than the correlation c'_{uni} resulting from applying the unifying intervention.

If it is more profitable for the advertiser to apply the antagonizing intervention, the difference $c'_{\text{ant}} - c'_{\text{uni}}$ can be interpreted as an externality imposed on the society:

Definition 5.4.14 (Polarization Cost)**.** *We define the polarization cost, denoted as ρ, as the difference of correlation between two agents after applying an antagonizing intervention and a unifying intervention, i.e., $\rho := c'_{uni} - c'_{ant}$.*

Figure 5-2 illustrates the polarization cost as a function of initial correlation c. Since the projection of the antagonizing intervention is parallel to u_1, it always achieves the same outcome that can be computed to be $u'_{1,3} = 1/3$. The difference between 1/3 and the value of the blue curve is the difference in affinity of the first opinion u'_1 in case of antagonizing and unifying interventions. It clearly seems that in certain situations this difference is large enough to justify applying the antagonizing strategy, increasing the polarization of opinions.

More detailed computations are contained in Section 5.8.1. Additionally, in Section 5.8.2 we discuss the problem of finding an optimal intervention with a larger number of opinions. We assume that there are n opinions $u_1, \ldots, u_n$ with $u_{i,d} = 0$ and that we want to find

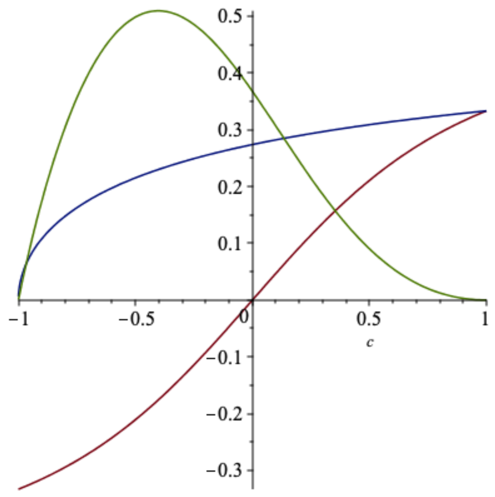


Figure 5-2: *Metrics of performance in our example. The blue line shows the affinity $u'_{1,3} = u'_{2,3}$ after applying the unifying intervention. The red line shows affinity $u'_{2,3}$ of the second agent for the antagonizing intervention (recall that $u'_{1,3} = 1/3$). The green line shows the polarization cost $\rho = c_{\text{uni}} - c_{\text{ant}}$.*

an intervention maximizing the number of agents with the final opinion $u'_{i,d} > T$ for some $0 < T < 1$.

Interestingly, we show that this problem is equivalent to a generalization of the densest hemisphere problem from the long-term scenario discussed in Section 5.4.2. More precisely, it is equivalent to finding a densest *spherical cap* of a given radius (that depends on the threshold T), see Proposition 5.8.1 in Section 5.8.2.

5.4.4 Two influencers: two randomized interventions polarize

Finally, we analyze a scenario where there are two influencers with differing agendas, represented by different[1] intervention vectors v and v'. We consider the *randomized* setup, where at each time step, one of the influencers is randomly chosen to apply their intervention. We demonstrate that this setting also results, in most cases and in a certain sense, in the polarization of agents.

Recall that a convex cone of two vectors v and v' is the set $\{\alpha v + \beta v' : \alpha, \beta \geq 0\}$. A precise

[1] We also assume that $v \neq -v'$, as otherwise the intervention effects are the same in our model.

statement that we prove is:

Theorem 5.4.15. *Let $\langle v, v'\rangle > 0$ and let a starting opinion $u^{(1)}$ be such that $\langle u^{(1)}, v\rangle \neq 0$ or $\langle u^{(1)}, v'\rangle \neq 0$. Then, as t goes to infinity and almost surely, either the Euclidean distance between $u^{(t)}$ and the convex cone generated by v and v′ or between $u^{(t)}$ and the convex cone generated by $-v$ and $-v'$ goes to 0.*

In order to justify the assumptions of Theorem 5.4.15, note that if an agent starts with an opinion u such that

$$\langle u, v\rangle = \langle u, v'\rangle = 0\,, \tag{5.7}$$

applying v or v' never changes their opinion. In Theorem 5.4.15 we show that if (5.7) does not hold and, additionally, $\langle v, v'\rangle \neq 0$, (if $\langle v, v'\rangle < 0$ we can exchange v' with $-v'$ without changing the effects of any interventions), the opinion vector with probability 1 ends up either converging to the convex cone generated by v and v' or the convex cone generated by $-v$ and $-v'$. In particular, since vectors u for which (5.7) holds form a set of measure 0, if n initial opinions are sampled i.i.d. from an absolutely continuous distribution, almost surely all opinions converge to the convex cones (which are themselves sets of measure 0).

Furthermore, we attempt to strengthen this notion of polarization. As in Theorem 5.4.1, the best we can hope for is that for each pair of opinions either the distance between $u_1^{(t)}$ and $u_2^{(t)}$ or between $u_1^{(t)}$ and $-u_2^{(t)}$ converges to 0. Letting $V := \mathrm{span}\{v, v'\}$ and $W := V^\perp$ and writing any vector u as a sum of its projections $u = u_V + u_W$, we show:

Theorem 5.4.16. *Suppose that $\langle v, v'\rangle > 1/\sqrt{2+\eta}$ and let $u_1^{(1)}, u_2^{(1)}$ be such that $\|(u_1^{(1)})_V\| \neq 0$, $\|(u_2^{(1)})_V\| \neq 0$. Then, almost surely, either $\|u_1^{(t)} - u_2^{(t)}\|$ converges to 0, or $\|u_1^{(t)} + u_2^{(t)}\|$ converges to 0.*

In other words, we prove a stronger notion of convergence in case the correlation between interventions v and v' is larger than

$$\langle v, v'\rangle > \sqrt{\frac{1}{2+\eta}} > \frac{\sqrt{2}}{2} \approx 0.71\,.$$

In particular, for $\eta = 1$ our result applies if $\langle v, v' \rangle > \sqrt{3}/3 \approx 0.58$. Our experiments suggest that this convergence occurs also for other values of $\langle v, v' \rangle$, but we do not prove it here.

In the remaining case when v and v' are orthogonal, there is no natural notion of a convex cone (the vectors v and v' form four right angles), but we can still show that an initial opinion $u^{(1)}$ converges to the *quadrant* in which it starts with respect to v and v'. Namely, for all t, we have that $\operatorname{sgn}\left(\left\langle u^{(t)}, v \right\rangle\right) = \operatorname{sgn}\left(\left\langle u^{(1)}, v \right\rangle\right)$ and $\operatorname{sgn}\left(\left\langle u^{(t)}, v' \right\rangle\right) = \operatorname{sgn}\left(\left\langle u^{(1)}, v' \right\rangle\right)$, and furthermore the distance between $u^{(t)}$ and the subspace V goes to 0 with t:

Corollary 5.4.17. *Let $\langle v, v' \rangle = 0$ and let an initial opinion $u = u^{(1)}$ be such that $\langle u, v \rangle \neq 0$ and $\langle u, v' \rangle \neq 0$. Then, almost surely, the following facts hold:*

1. *$\|u_W^{(t)}\| \to 0$ as $t \to \infty$.*

2. *For all t, $\operatorname{sgn}\left(\left\langle u^{(t)}, v \right\rangle\right) = \operatorname{sgn}\left(\left\langle u^{(1)}, v \right\rangle\right)$ and $\operatorname{sgn}\left(\left\langle u^{(t)}, v' \right\rangle\right) = \operatorname{sgn}\left(\left\langle u^{(1)}, v' \right\rangle\right)$.*

The proofs of Theorems 5.4.15 and 5.4.16 and Corollary 5.4.17 are contained in Section 5.9.

5.5 Illustrative Examples

In this section we give some illustrative examples of the dynamics of our model, by using simulations with η in Equation 5.1 set to 1.

5.5.1 One advertiser

To illustrate our model, suppose an advertiser is marketing a new product. The opinion of the population has four dimensions. The population consists of 500 agents, each with random initial opinions $u_i = (u_{i,1}, u_{i,2}, u_{i,3}, 0)$ subject to $u_{i,1}^2 + u_{i,2}^2 + u_{i,3}^2 = 1$. The opinion on the new product is represented by the fourth coordinate, which is initially set to zero for all agents.

Suppose the advertiser chooses a strategy of repeatedly applying an intervention that couples the product with the preexisting opinion on the first coordinate. For example, an intervention vector could be $v = (\sqrt{1-\alpha^2}, 0, 0, \alpha)$ for $\alpha = 3/4$.

The evolution of opinions over five consecutive broadcasts of v is illustrated in Figure 5-3. The interventions increase the affinity for the product for some agents while antagonizing others. Furthermore, the interventions have a side effect in that the agents' opinions on the first three coordinates also become polarized.

5.5.2 Two advertisers

For another slightly more involved example, suppose there are two advertisers marketing their products. Agents' opinions now have five dimensions ($d = 5$) with the fourth and fifth coordinates corresponding to the opinions on these two products. Initially, 500 opinions on the first three coordinates are distributed randomly and uniformly on a three-dimensional sphere, and the last two coordinates are equal to zero.

Suppose the two advertisers apply interventions v_1 and v_2 in an alternating fashion. We take v_1 and v_2 to be orthogonal. For example, let

$$v_1 = (\sqrt{1-\alpha^2}, 0, 0, \alpha, 0)$$
$$v_2 = (0, \sqrt{1-\alpha^2}, 0, 0, \alpha)$$

for $\alpha = 3/4$.

In Figure 5-4 we illustrate the agents' opinions after each advertiser applied their intervention two, four and six times (so the total of, respectively, four, eight and twelve interventions have been applied). A pattern of polarization on the fourth and fifth coordinates can be observed. At the same time, the pattern on the first three coordinates is more complicated: The opinions on these dimensions are scattered around a circle on the plane spanned by the first two coordinates. This is a somewhat special behavior that arises because vectors v_1 and v_2 are orthogonal. It is connected to the difference between

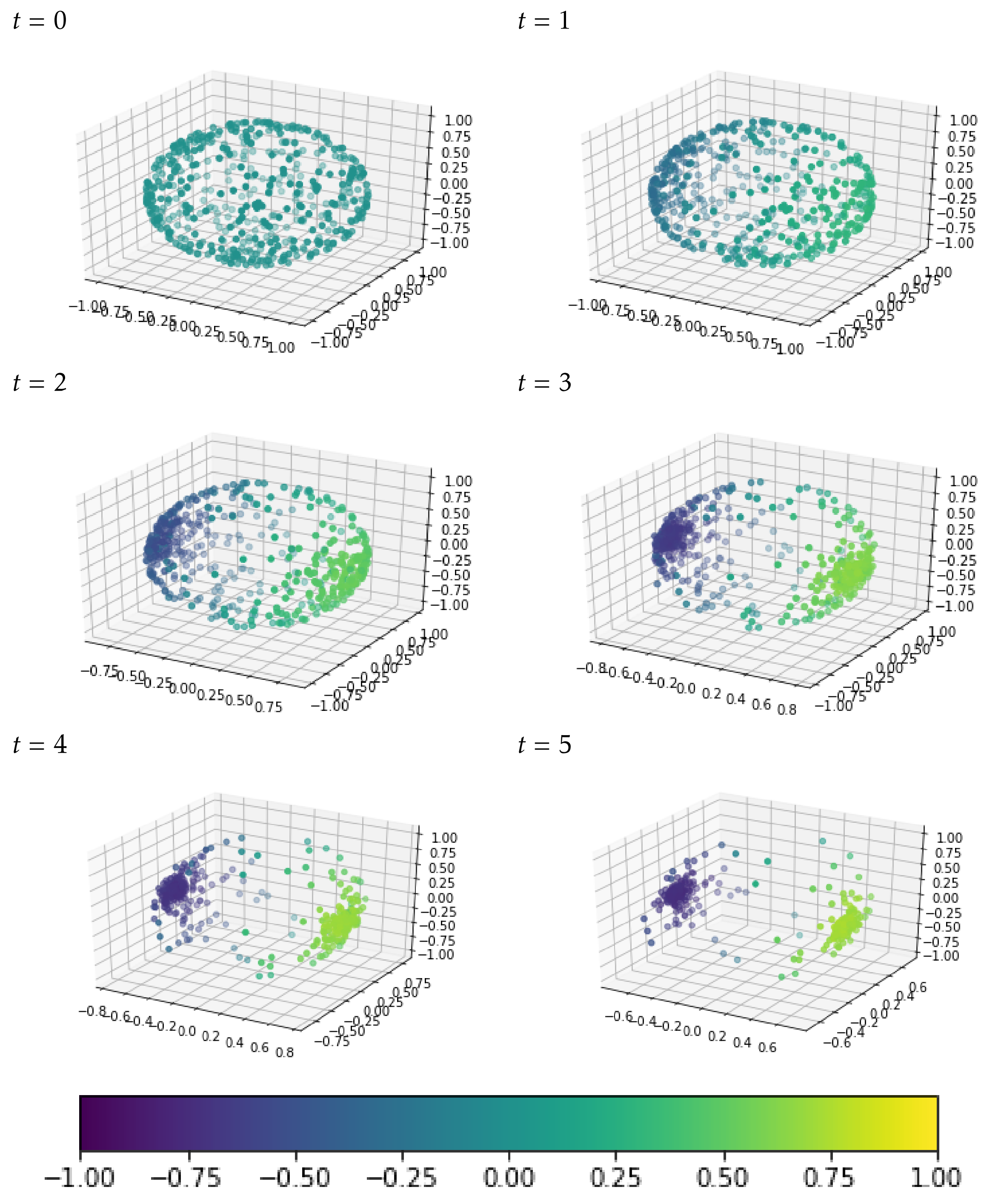


Figure 5-3: *Five consecutive interventions of vector $v = (\sqrt{7}/4, 0, 0, 3/4)$ in $\mathbb{R}^4$. The spatial position of the points represents their first three coordinates. The fourth dimension, representing the marketed product, is encoded with a color scale.*

Theorem 5.4.15 and Corollary 5.4.17 discussed in Section 5.4.4.

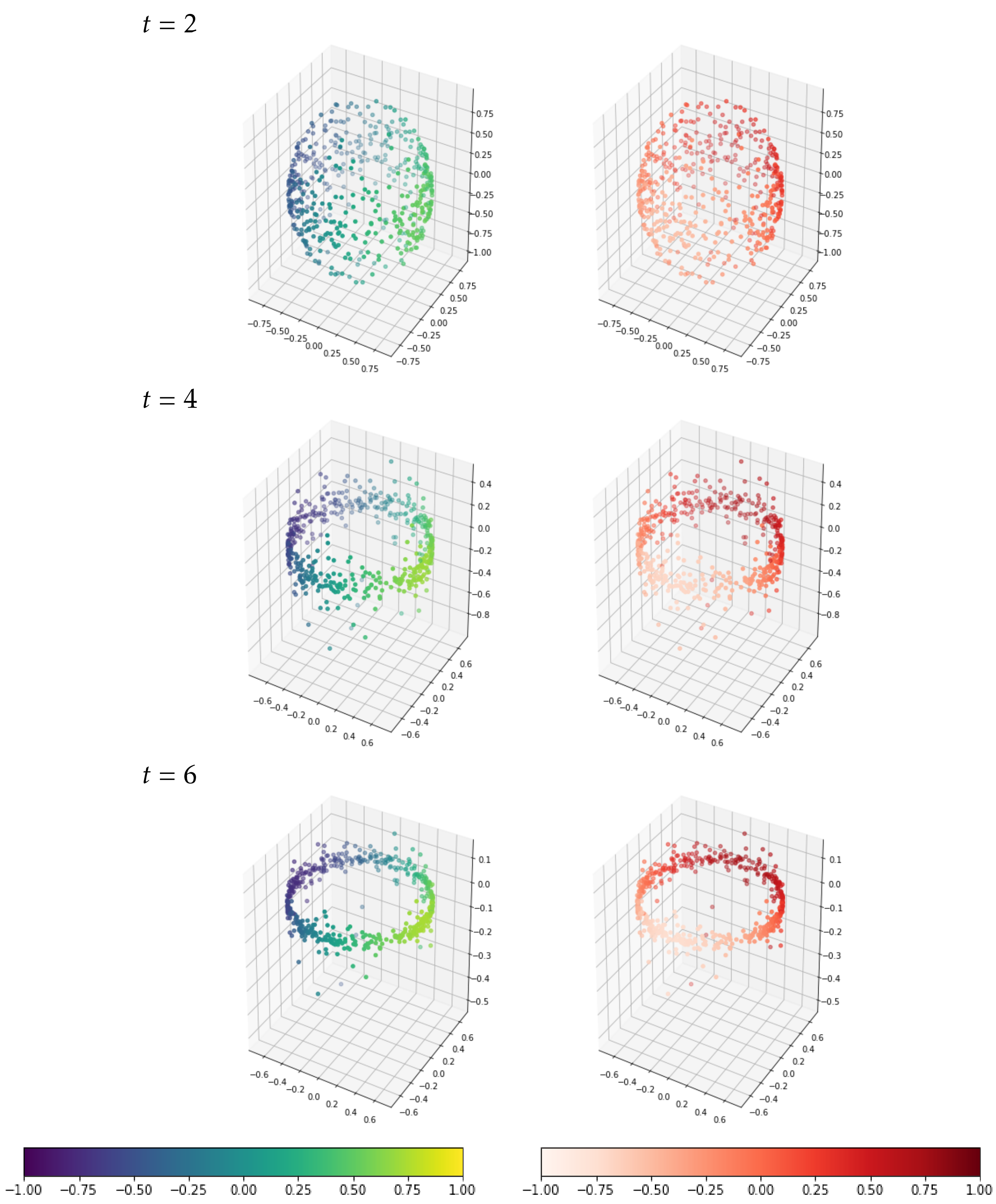


Figure 5-4: *Two figures are displayed for each time step $t = 2, 4, 6$. The spatial positions of the points in both figures correspond to the first three dimensions (they are the same on the left and right). The colors encode opinions of the two products. The left column presents population's opinions on the first product (fourth coordinate). The right column presents the population's opinions on the second product (fifth coordinate). The distribution of points at $t = 0$ is uniform (cf. Figure 5-3).*

5.6 Long-Term Polarization of Random Interventions

Let us explain the outline of the proof of Theorem 5.4.1. We consider a random variable $\alpha_t \in [0,\pi]$ representing the angle between $u_1^{(t)}$ and $u_2^{(t)}$ and we show that it is a martingale. By the martingale convergence theorem, this means that α_t almost surely converges. Furthermore, since for $0 < \varepsilon < \alpha_t < \pi - \varepsilon$ the conditional variance $\mathrm{Var}[\alpha_t \mid \alpha_{t-1}]$ is bounded away from 0, the only possible convergence points are 0 and π. Below we develop this idea in more detail.

We also remark that for $d \geq 3$ simulations suggest that Theorem 5.4.1 is still true, however our proof fails since α_t ceases to be a martingale.

As for Corollary 5.4.2, it follows from Theorem 5.4.1 by applying the union bound (with probability 0 in each term) for each pair of opinions $u_i^{(1)}, u_j^{(1)}$.

Proof of Theorem 5.4.1. To start with, we develop some notation. Let $f : S^1 \times S^1 \to S^1$ be the function mapping an opinion u and an intervention v to an updated opinion $f(u,v)$, according to (5.2) and (5.3). Note that this function is invariant under rotation: namely, for any real unitary transformation $A : S^1 \to S^1$ we have

$$f(Au, Av) = Af(u,v)\,. \tag{5.8}$$

We will now state and prove two claims. In both of them we fix a time t and opinions $u := u_1^{(t)}$, $u' := u_2^{(t)}$. We also let $v := v^{(t)}$. As discussed, define

$$\alpha_t := \arccos\langle u, u'\rangle \in [0,\pi]$$

as the primary angle between u and u'. One consequence of (5.8) is that we can assume wlog that $u = (1,0)$ and $u' = (\cos\alpha_t, \sin\alpha_t)$.

Claim 5.6.1. $\mathrm{E}[\alpha_{t+1} \mid \alpha_t] = \alpha_t$.

Proof. Let us write the random intervention vector as $v = (\cos\beta, \sin\beta)$, where the distribu-

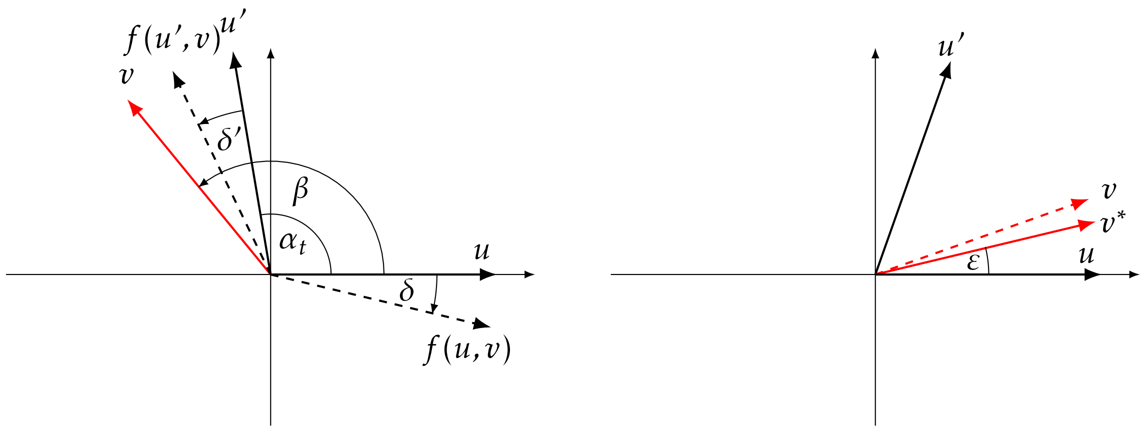


Figure 5-5: *On the left an illustration of the vectors and angles in the proof of Claim 5.6.1. On the right an illustration for the proof of Claim 5.6.3.*

tion of β is uniform in $[0, 2\pi)$. We will also write (cf. Figure 5-5 for an overview)

$$f(u,v) = (\cos\delta, \sin\delta)\,, \qquad f(u',v) = (\cos(\alpha_t+\delta'), \sin(\alpha_t+\delta'))\,, \qquad \delta,\delta' \in [-\pi,\pi)\,.$$

Since $f(u,-v) = f(u,v)$ and since if $\langle u,v\rangle \geq 0$, then $f(u,v)$ is a convex combination of u and v, we conclude that an intervention cannot move the angle of vector u by more than $\pi/2$: in other words, we have $-\pi/2 < \delta,\delta' < \pi/2$. Furthermore, since it is easy to check that $\delta(\beta) = -\delta(2\pi-\beta)$, we also have

$$\int_0^{2\pi} \delta \,\mathrm{d}\beta = 0\,, \tag{5.9}$$

and, similarly, applying (5.8), $\int_0^{2\pi} \delta' \,\mathrm{d}\beta = 0$.

Let $\alpha^* := \alpha_t + \delta' - \delta \pmod{2\pi}$, i.e., we take α^* to be the directed angle from $f(u,v)$ to $f(u',v)$ reduced to lie in the interval $[0,2\pi)$ A crucial observation is that

$$0 \leq \alpha^* \leq \pi\,. \tag{5.10}$$

This is a result of our choice of the coordinate system and the fact that applying an intervention does not change the orientation of opinion vectors. Let us proceed further, assuming (5.10) and deferring its proof until later.

Due to (5.10), we get

$$\alpha_{t+1} = \arccos\langle f(u,v), f(u',v)\rangle = \arccos(\cos(\alpha_t + \delta')\cos\delta + \sin(\alpha_t + \delta')\sin\delta) = \arccos(\cos(\alpha^*)) = \alpha^* .$$

We can now start substituting

$$\mathrm{E}[\alpha_{t+1} \mid \alpha_t] = \frac{1}{2\pi}\int_0^{2\pi} \alpha_{t+1}\,\mathrm{d}\beta = \frac{1}{2\pi}\int_0^{2\pi} \alpha^*\,\mathrm{d}\beta = \frac{1}{2\pi}\int_0^{2\pi} \alpha_t + \delta' - \delta \pmod{2\pi}\,\mathrm{d}\beta . \quad (5.11)$$

We now argue that

$$\alpha_t + \delta' - \delta \pmod{2\pi} = \alpha_t + \delta' - \delta , \quad (5.12)$$

that is $0 \le \alpha_t + \delta' - \delta < 2\pi$. Indeed, using $0 \le \alpha_t \le \pi$ and $-\pi/2 < \delta, \delta' < \pi/2$ we have

$$-\pi < \alpha_t + \delta' - \delta < 2\pi ,$$

but $-\pi < \alpha_t + \delta' - \delta < 0$ would imply $\alpha^* > \pi$, contradicting (5.10). Finally, substituting (5.12) and (5.9) into (5.11), we arrive at

$$\mathrm{E}[\alpha_{t+1} \mid \alpha_t] = \alpha_t + \frac{1}{2\pi}\int_0^{2\pi} \delta'\,\mathrm{d}\beta - \frac{1}{2\pi}\int_0^{2\pi} \delta\,\mathrm{d}\beta = \alpha_t ,$$

concluding the proof. □

It remains to deal with (5.10):

Fact 5.6.2. $0 \le \alpha^* \le \pi$.

Proof. Let us embed our underlying space $\mathbb{R}^2$ in $\mathbb{R}^3$ by setting the last coordinate to zero. Letting $\times$ denote the cross product, we have

$$u \times u' = (0, 0, \sin\alpha_t) , \qquad f(u,v) \times f(u',v) = (0, 0, \sin\alpha^*) .$$

Since the case $\alpha_t \in \{0, \pi\}$ is easily handled by noticing that $\alpha^* = \alpha_t$, we can assume that $0 < \alpha_t < \pi$. In that case, it is enough that we prove

$$\langle u \times u', f(u,v) \times f(u',v) \rangle \geq 0\,. \tag{5.13}$$

Setting $C(w) := \sqrt{1 + (2\eta + \eta^2)\langle w, v\rangle^2}$, we apply (5.2) and bilinearity of cross product to compute

$$\begin{aligned} f(u,v) \times f(u',v) &= \frac{1}{C(u)C(u')}\Big(u \times u' + \eta\Big(\langle u,v\rangle(v \times u') + \langle u',v\rangle(u \times v)\Big)\Big) \\ &= \frac{1}{C(u)C(u')}\Big(u \times u' + \eta\Big(u \times u' + (\langle u,v\rangle v - u) \times (u' - \langle u',v\rangle v)\Big)\Big) &\quad (5.14) \\ &= \frac{1+\eta}{C(u)C(u')} u \times u'\,, &\quad (5.15) \end{aligned}$$

where in (5.14) we used the identity $a \times b + c \times d = a \times d + c \times b + (a-c) \times (b-d)$, and in (5.15) we used that both $\langle u,v\rangle v - u$ and $u' - \langle u',v\rangle v$ are projections of vectors onto the line orthogonal to v, and therefore they are parallel and their cross product vanishes.

Consequently, we can conclude that $f(u,v) \times f(u',v)$ is parallel to $u \times u'$ with a positive proportionality constant, which implies (5.13) and concludes the proof. $\square$

Claim 5.6.3. *For every $\varepsilon > 0$, there exists $\delta > 0$ such that,*

$$\varepsilon \leq \alpha_t \leq \pi/2 \implies \mathbb{P}\Big[\alpha_{t+1} < \alpha_t - \delta \mid \alpha_t\Big] > \delta\,, \tag{5.16}$$

and, symmetrically,

$$\pi/2 \leq \alpha_t \leq \pi - \varepsilon \implies \mathbb{P}\Big[\alpha_{t+1} > \alpha_t + \delta \mid \alpha_t\Big] > \delta\,. \tag{5.17}$$

Proof. Note that our intervention function f exhibits a symmetry $f(-u,v) = -f(u,v)$.

Furthermore, we also have $\arccos\langle u, u'\rangle = \pi - \arccos\langle u, -u'\rangle$. Consequently,

$$\begin{aligned}\alpha_{t+1} - \alpha_t &= \arccos\langle f(u,v), f(u',v)\rangle - \arccos\langle u, u'\rangle \\ &= \pi - \arccos\langle f(u,v), f(-u',v)\rangle - (\pi - \arccos\langle u, -u'\rangle) \\ &= -\Big(\arccos\langle f(u,v), f(-u',v)\rangle - \arccos\langle u, -u'\rangle\Big) .\end{aligned}$$

As a result, it is enough that we prove (5.16) and then (5.17) follows immediately by replacing u' with $-u'$.

Consider vector $v^* := (\cos\varepsilon, \sin\varepsilon)$ (see Figure 5-5). We will now show that if $\varepsilon \le \alpha_t \le \pi/2$ and the intervention v is sufficiently close to v^*, then v decreases the angle between u and u'. To that end, let us use a metric on S^1 given by

$$D(u,v) := \arccos\langle u, v\rangle .$$

Note that this metric is strongly equivalent to the standard Euclidean metric on S^1. We can now us triangle inequality to write

$$\begin{aligned}\alpha_{t+1} &= D(f(u,v), f(u',v)) \\ &\le D(f(u,v), f(u,v^*)) + D(f(u,v^*), v^*) + D(v^*, f(u',v^*)) + D(f(u',v^*), f(u',v)) . \quad (5.18)\end{aligned}$$

Let us now bound the terms in (5.18) one by one.

First, since, by (5.2), $f(u,v^*)$ is a strict convex combination of u and v^*, we have

$$D(f(u,v^*), v^*) = d(\varepsilon) < D(u,v^*) = \varepsilon .$$

Similarly,

$$D(v^*, f(u',v^*)) \le D(v^*, u') = \alpha_t - \varepsilon .$$

Second, since f is continuous, if we assume that $D(v,v^*) < \delta$, or equivalently, $\|v - v^*\| < \delta$

for small enough $\delta > 0$, then we can make both $D(f(u,v), f(u,v^*))$ and $D(f(u',v^*), f(u',v))$ as small as needed (for example, less than $(\varepsilon - d(\varepsilon))/4$).

All in all, we have that for some $\delta = \delta(\varepsilon) > 0$,

$$\begin{aligned}\|v - v^*\| < \delta \implies \alpha_{t+1} &< \frac{\varepsilon - d(\varepsilon)}{4} + d(\varepsilon) + (\alpha_t - \varepsilon) + \frac{\varepsilon - d(\varepsilon)}{4} \\ &= \alpha_t - \frac{\varepsilon - d(\varepsilon)}{2} \, .\end{aligned}$$

However, clearly, $\mathbb{P}[\|v - v^*\| < \delta] = p(\varepsilon) > 0$. Therefore, taking $\delta' := \min\Big(p(\varepsilon), (\varepsilon - d(\varepsilon))/2\Big)$, we have

$$\mathbb{P}[\alpha_{t+1} < \alpha_t - \delta' \mid \alpha_t] > \delta' \, ,$$

as claimed in (5.16). □

As a consequence of applying Claim 5.6.3 $\lceil \pi/\delta \rceil$ times, we obtain that for every $\varepsilon > 0$ there exist $k \in \mathbb{N}$ and $\eta < 1$ such that

$$\varepsilon \le \alpha_t \le \pi - \varepsilon \implies \mathbb{P}[\varepsilon \le \alpha_{t+k} \le \pi - \varepsilon \mid \alpha_t] \le \eta \, .$$

Subsequently, it follows that for any fixed $\varepsilon > 0$ and $T \in \mathbb{N}$,

$$\mathbb{P}[\forall t \ge T : \varepsilon \le \alpha_{T+t} \le \pi - \varepsilon] = 0 \, . \tag{5.19}$$

To finish the proof of Theorem 5.4.1, we use standard tools from theory of martingales. By Claim 5.6.1, the sequence of random variables α_t is a bounded martingale and therefore almost surely converges. Accordingly, let $\alpha^* := \lim_{t\to\infty} \alpha_t$. To finish the proof, we need to show that $\mathbb{P}[0 < \alpha^* < \pi] = 0$. To that end,

$$\begin{aligned}\mathbb{P}[0 < \alpha^* < \pi] &\le \sum_{s=1}^{\infty} \mathbb{P}\left[\frac{1}{s} < \alpha^* < \pi - \frac{1}{s}\right] \le \sum_{s=1}^{\infty} \mathbb{P}\left[\exists T : \forall t \ge T : \frac{1}{2s} < \alpha_t < \pi - \frac{1}{2s}\right] \\ &\le \sum_{s=1}^{\infty}\sum_{T=1}^{\infty} \mathbb{P}\left[\forall t \ge T : \frac{1}{2s} < \alpha_t < \pi - \frac{1}{2s}\right] = 0\,,\end{aligned}$$

where we applied (5.19) in the last line. □

5.7 Long-Term Strategy: Computational Aspects

In this section we provide proof of Theorem 5.4.11.

5.7.1 Computational equivalence to learning halfspaces

By Theorem 5.4.3, computing an optimal strategy to get the most people to agree with an opinion v^* is equivalent to computing the hemisphere maximizing the number of agent opinions inside. In this section we discuss the computational consequences of this fact following from known results. It turns out that the densest hemisphere problem is closely related to finding a maximum agreement halfspace, studied in the context of learning halfspaces in perceptron problems. This problem can be stated as follows:

Definition 5.7.1 (Maximum Agreement Halfspace)**.** *In the problem of* maximum agreement halfspace, *given a labeled set of points $D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^d \times \{\pm 1\}$, the objective is to find a halfspace $H = \{x : \langle v, x\rangle > c\}$ maximizing the* agreement

$$A(D, H) = \frac{\sum_{i=1}^{n} \mathbb{1}[y_i \cdot x_i \in H]}{n}\,.$$

As pointed out in [20], there exists a reduction from the maximum agreement halfspace problem to the densest hemisphere problem that preserves the quality of solutions. Since this reduction is only briefly sketched in [20], we describe it below.

The reduction proceeds as follows: Given a labeled set $D = \{(x_1, y_1), \ldots, (x_n, y_n)\} \in \mathbb{R}^d \times \{\pm 1\}$, we map it to $D' = \{x'_1, \ldots, x'_n\} \in \mathbb{R}^{d+1}$ using the formula

$$x'_i = \frac{y_i}{\sqrt{1 + \|x_i\|^2}} \cdot (x_i, 1) \,.$$

In other words, we proceed in three steps: First, we add a coordinate and set its value to 1 for every point x_i. Second, we normalize each resulting point so that it lies on the unit sphere in $\mathbb{R}^{d+1}$. Finally, we negate each point that came with negative label $y_i = -1$.

This is a so-called "strict reduction", which is expressed in the following claim:

Claim 5.7.2. *The solutions (halfspaces) for an instance of Maximum Agreement Halfspace D are in one-to-one correspondence with solutions (hemispheres) for the reduced instance of Densest Hemisphere D'. Furthermore, for a corresponding pair of solutions (H, H') the agreement $A(D, H)$ is equal to the density $|D' \cap H'|/n$.*

Proof. It is more convenient to think of solutions for D' as homogeneous, open halfspaces $H' = \{x \in \mathbb{R}^{d+1} : \langle v, x \rangle > 0\}$.

With that in mind, we map a solution to the maximum agreement halfspace problem $H = \{x \in \mathbb{R}^d : \langle v, x \rangle > c\}$ to a solution to the densest hemisphere problem $H' = \{(x, x_{d+1}) \in \mathbb{R}^{d+1} : \langle (v, -c)(x, x_{d+1}) \rangle > 0\}$. Clearly, this is a one-to-one mapping between open halfspaces in $\mathbb{R}^d$ and homogeneous open halfspaces in $\mathbb{R}^{d+1}$.

Furthermore, it is easy to verify that $y_i \cdot x_i \in H$ if and only if $x'_i \in H'$ and therefore $A(D, H) = |D' \cap H'|/n$. $\square$

The reduction allows us to use a strong hardness of approximation result from [68] (see also [47, 30, 18, 7] for related work):

Theorem 5.7.3 ([68])**.** *Unless P=NP, for any $\varepsilon > 0$, there is no polynomial time algorithm that distinguishes the following, given an instance of maximum agreement halfspace problem:*

- *There exists a halfspace H such that $A(D, H) > 1 - \varepsilon$.*
- *For every halfspace H we have $A(D, H) < 1/2 + \varepsilon$.*

In other words, it is computationally hard to distinguish between instances that have halfspaces with almost perfect agreement and instances where there is no halfspace with agreement noticeable larger than 1/2 (of course for any hyperplane, one of the two halfspaces defined by this hyperplane has an agreement at least 1/2). Consequently, unless P=NP, there is no polynomial time algorithm that, for any $\varepsilon > 0$, given an instance that has a halfspace with agreement $1-\varepsilon$, finds a halfspace with agreement more than $1/2+\varepsilon$. We note that the results in [68] show hardness for instances with dimension d comparable to the number of points n.

Finally, by standard (and straightforward) arguments from complexity theory, Theorem 5.4.11 follows from Theorem 5.7.3 and Claim 5.7.2.

5.8 Short-Term Strategies

5.8.1 One intervention, two agents: polarization costs

As discussed, it is enough to consider only three dimensions $d = 3$ with the initial opinions given by (5.6). First, consider the antagonizing intervention where the influencer maximizes their appeal to the first agent. Clearly, the intervention should be of the form

$$v = \cos\beta \cdot u_1 + \sin\beta \cdot (0,0,1)$$

for some $0 \le \beta \le \pi/2$. Substituting in (5.2), we compute

$$(u'_{1,3})^2 = \frac{\cos^2\beta\sin^2\beta}{1+3\cos^2\beta}\,. \tag{5.20}$$

Maximizing (5.20), we get $\cos\beta = \sqrt{3}/3$ and $u'_{1,3} = 1/3$. The value 1/3 is the benchmark for what can be achieved by a single intervention: It is a maximum value for $u'_{1,3}$ attainable provided that initially $u_{1,3} = 0$.

What is the effect of this strategy on the other opinion u_2? Again substituting into (5.2), we get

$$u'_{2,3} = \frac{c\sqrt{2}}{3\sqrt{1+c^2}}\,.$$

The graph $v_{2,3}$ as a function of the correlation $c \in [-1,1]$ is shown in red in Figure 5-2. In particular, the graph increases from $-1/3$ to $1/3$, passing through 0 for $c = 0$.

Moving to the unifying strategy, in this case it is not difficult to see (cf. Figure 5-1) that the intervention vector should be of the form

$$v = (0, \cos\beta, \sin\beta)$$

for some $0 \le \beta \le \pi/2$. A computation in computer algebra system (CAS) establishes that $v_{1,3} = v_{2,3}$ is maximized for

$$\cos^2\beta = \frac{\sqrt{2}(\sqrt{3c+5}-\sqrt{2})}{3(c+1)}\,,$$

yielding a somewhat complicated expression

$$u'_{1,3} = u'_{2,3} = \sqrt{\frac{3c+7-2\sqrt{6c+10}}{9(c+1)}}\,.$$

This function is depicted in Figure 5-2 in blue. In particular, for $c \in [-1,1]$, it increases from 0 to 1/3 and its value at 0 is approximately 0.27. Furthermore, its growth close to $c = -1$ is of the square-root type.

If the influencer chooses the unifying intervention, the correlation $c' = \langle u'_1, u'_2 \rangle$ increases in comparison to the initial correlation c. On the other hand, in case of antagonizing intervetnion, if the initial correlation is negative $c < 0$, the correlation after intervention will decrease, increasing the polarization of opinions. Let us denote the correlations $\langle u'_1, u'_2 \rangle$ in the unifying and antagonizing cases as, respectively, c'_{uni} and c'_{ant}. Another CAS computation gives

$$c'_{\text{ant}} = \frac{c\sqrt{2}}{\sqrt{c^2+1}} \,,$$
$$c'_{\text{uni}} = 1 - \frac{\sqrt{2}(1-c)}{\sqrt{3c+5}} \,.$$

The polarization cost $c'_{\text{uni}} - c'_{\text{ant}}$ is shown in Figure 5-2 in green.

Looking at Figure 5-2 one can draw some qualitative conclusions about consequences of unifying and antagonizing strategies for different values of initial correlation c. For example, if the inital opinions are uncorrelated ($c = 0$), the unifying strategy gives

$$u'_{1,3} = u'_{2,3} \approx 0.27 \,, \quad c'_{\text{uni}} \approx 0.37 \,,$$

while the antagonizing strategy has

$$u'_{1,3} = 1/3 \,, \quad u'_{2,3} = 0 \,, \quad c'_{\text{ant}} = 0 \,.$$

Depending on their incentives, the influencer might choose the antagonizing strategy, forgoing a chance for a substantial increase in the agreement among the agents.

A more pronounced case of high polarization cost occurs if the initial opinions are already substantially polarized. For example, for $c = -0.7$ the unifying strategy has

$$u'_{1,3} = u'_{2,3} \approx 0.18 \,, \quad c'_{\text{uni}} \approx -0.41 \,,$$

while the antagonizing strategy gives

$$u'_{1,3} = 1/3 \,, \quad u'_{2,3} \approx -0.27 \,, \quad c'_{\text{ant}} \approx -0.81 \,.$$

Since the difference in $u'_{1,3}$ is more pronounced, the influencer might have more incentive to apply antagonizing strategy resulting in a high polarization cost.

On the other hand, the polarization cost is low if the initial correlation c is either high or low. If c is close to 1, then there is not much difference between unifying and antagonizing interventions. On the other hand, if c is close to -1, then neither strategy changes the correlation much, while the unifying strategy has also little effect on the agents' opinions of the product.

5.8.2 One intervention, many agents: finding the densest spherical cap

A more general version of the problem of persuading with limited number of interventions features n agents with opinions $u_1, \dots, u_n \in \mathbb{R}^d$. The influencer is given a threshold $0 < T < 1$ and can apply one intervention v with the objective of maximizing the number of agents such that $u'_{i,d} \geq T$. The value T can be interpreted as a threshold above which a consumer decides to buy the advertised product, or more generally take a desired action, such as go to vote, donate, etc.

Assume that the agents are initially agnostic about the product ($u_{i,d} = 0$). In that case we can also assume $T \leq 1/3$, since $1/3$ is the maximum value that can be achieved in the d-th coordinate by a single intervention, cf. (5.20). It turns out an analogy to the densest hemisphere problem can be observed. We show that after fixing the threshold T, the problem becomes equivalent to finding a spherical cap of a given radius in $d-1$ dimensions that contains the maximum number of agent opinions. To state our result, let us abuse notation and write vectors $u \in \mathbb{R}^d$ as $u = (u^*, u_d)$ for $u^* \in \mathbb{R}^{d-1}$, $u_d \in \mathbb{R}$:

Proposition 5.8.1. *In the setting above, let*

$$c := \frac{2T}{1-3T^2}\,, \qquad z := \frac{\sqrt{\sqrt{1+3c^2}-1}}{\sqrt{3}c}\,, \qquad \beta := \arccos(z)\,. \tag{5.21}$$

Then, the number of agents with $u'_{i,d} \geq T$ is maximized by applying an intervention

$$v := (\cos\beta \cdot v^*, \sin\beta)$$

for $v^* \in \mathbb{R}^{d-1}$ that maximizes the number of agents satisfying

$$\langle u_i^*, v^* \rangle \geq c\,.$$

Proof. Let us write a generic intervention vector as

$$v = (\cos\beta \cdot v^*, \sin\beta)\,,$$

where $0 \leq \beta \leq \pi/2, v^* \in \mathbb{R}^{d-1}$ and $\|v^*\| = 1$. If v is applied to an opinion vector $u_i = (u_i^*, 0)$ and we let $c_i := \langle u_i^*, v^* \rangle$, substituting into (5.2) we can compute

$$u_i + \langle u_i, v \rangle \cdot v = (u_i^* + c_i \cos^2\beta \cdot v^*, c_i \cos\beta \sin\beta)\,,$$

and therefore, using (5.4),

$$u'_{i,d} = \frac{c_i \cos\beta \sin\beta}{\sqrt{1 + 3c_i^2 \cos^2\beta}} = \frac{c_i z \sqrt{1 - z^2}}{\sqrt{1 + 3c_i^2 z^2}}\,, \tag{5.22}$$

where we let $z := \cos\beta$.

Consider a fixed direction $v^* \in \mathbb{R}^{d-1}$. In order to maximize $u'_{i,d}$ for a point u_i with $\langle u_i^*, v^* \rangle = c$, we need to optimize over z in (5.22), resulting in $z = \sqrt{\sqrt{1 + 3c^2} - 1}/(\sqrt{3}c)$ and, substituting,

$$u'_{i,d} = \frac{\sqrt{1 + 3c^2} - 1}{3c}\,. \tag{5.23}$$

Note that from the definitions it is clear that the right-hand side of (5.22) is increasing in c_i for a fixed z. Therefore, in order to maximize the number of points with $u'_{i,d} \geq T$ for a fixed v^*, we should solve the equation $T = \frac{\sqrt{1+3c^2}-1}{3c}$ for c, resulting in $c = \frac{2T}{1-3T^2}$ and apply the intervention

$$v = (\cos\beta \cdot v^*, \sin\beta)\,,$$

just as claimed in (5.21). This intervention will succeed for all opinions satisfying

$$u'_{i,d} \geq T \iff \langle u_i^*, v^* \rangle \geq c \, ,$$

which means that the points u_i^* on which the objective $u'_{i,d} \geq T$ is achieved are exactly those contained in the spherical cap $\{x \in \mathbb{R}^{d-1} : \langle x, v^* \rangle \geq c\}$. Maximizing over all $v^* \in \mathbb{R}^{d-1}$ completes the proof. □

Note that the solution to this short-term problem for T going to zero approaches the densest hemisphere solution to the long-term problem discussed in Section 5.4.2.

5.9 Asymptotic Effects of Two Dueling Influencers

In this section we provide proofs of Theorems 5.4.15 and 5.4.16 and Corollary 5.4.17. In the following we will always write $\langle v, v' \rangle = \cos(\theta)$ for $0 \leq \theta \leq \pi/2$.

5.9.1 Proofs of Theorem 5.4.15 and Corollary 5.4.17

Proof outline of Theorem 5.4.15 First, we show that the distance between $u^{(t)}$ and V almost surely goes to 0 as $t \to \infty$, by showing that the norm of the projection of $u^{(t)}$ onto W converges to 0. This is proved in Proposition 5.9.1.

Then, we demonstrate that the convex cone spanned by v and v' is absorbing: when the projection of $u^{(T)}$ onto V falls in the cone, then the projections of $u^{(t)}$ for $t \geq T$ always stay in the cone as well. This is proved in Proposition 5.9.2.

Finally, in Proposition 5.9.3 we show that almost surely the projection of $u^{(t)}$ onto V eventually enters either the cone spanned by v and v', or the cone spanned by $-v$ and $-v'$. More concretely, we show that at any time t, there is a sequence of T interventions that lands the projection of $u^{(t+T)}$ in one of the cones, for some T that is independent of t. Since this sequence occurs with probability 2^{-T}, which is independent of t, the opinion almost surely eventually enters one of the cones.

Proposition 5.9.1. *Let $\langle v, v' \rangle \geq 0$ and take an opinion vector u such that $\|u_V\| = c \geq 0$. Furthermore, let u' be the vector resulting from randomly intervening on u with either v or v'. Then:*

1. *$\|u'_W\|^2 \leq \|u_W\|^2$.*

2. *With probability at least 1/2, $\|u'_W\|^2 \leq \|u_W\|^2 \cdot (1 - (\eta^2/2 + \eta) \cdot c^2\theta^2/16)$.*

Proof. Recall from (5.2)–(5.4) that if $v^* \in \{v, v'\}$ is the intervention vector, then

$$u' = \alpha(u + \eta \langle u, v^* \rangle \cdot v^*)$$

where $\alpha = \sqrt{\frac{1}{1+(2\eta+\eta^2)\cdot\langle u, v^*\rangle^2}}$ is the normalizing constant. Observe that when we project onto W, the component in the direction of v^* vanishes, so we have that

$$u'_W = \alpha \cdot u_W \,,$$

and the first claim easily follows since $\alpha \leq 1$.

To establish the second point, we need to show that with probability 1/2 we have $\alpha^2 < 1$ or, equivalently, $\langle u, v^* \rangle^2 = \langle u_V, v^* \rangle^2 > 0$. If $\theta > 0$, the projected vector u_V cannot be orthogonal both to v and v' (cf. Figure 5-6). More precisely, for at least one of $v^* \in \{v, v'\}$ the primary angle between u and v^* (or $-v^*$) must be at most $\pi/2 - \theta/2$ and consequently

$$|\langle u_V, v^* \rangle| \geq \|u_V\| \cdot |\cos(\pi/2 - \theta/2)| \geq c \cdot \theta/4 \,,$$

resulting in

$$\alpha^2 = \frac{1}{1 + (2\eta + \eta^2) \cdot \langle u_V, v^* \rangle^2} \leq 1 - (\eta + \eta^2/2) \cdot \frac{c^2\theta^2}{16} \,.$$

□

Proposition 5.9.2. *Let $\langle v, v' \rangle \geq 0$ and take u to be an opinion vector and u' to be a vector resulting from intervening on u with either v or v'. If u_V is a conical combination of v and v', then also u'_V is such a conical combination.*

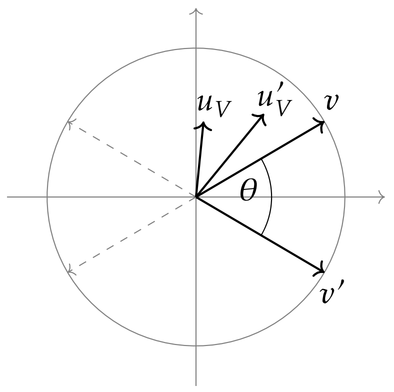


Figure 5-6: *Projection onto the subspace* $V = \mathrm{span}\{v, v'\}$.

Proof. Assume wlog that the vector applied is v. Then,

$$u'/\alpha = u + \eta \cdot \langle u, v\rangle \cdot v = u_V + \eta \cdot \langle u_V, v\rangle \cdot v + u_W \ .$$

Therefore, u'_V can be written as a nonnegative linear combination of u_V and v, where we use the fact that $\langle u_V, v\rangle$ is nonnegative, which follows since u_V is a conical combination of v and v', and $\langle v, v'\rangle \geq 0$. □

Next, we prove that when $\langle v, v'\rangle > 0$, the opinion u_t not only approaches subspace V, but also a specific area of V, namely, either $\mathrm{cone}(v, v')$ or $\mathrm{cone}(-v, -v')$.

Proposition 5.9.3. *Let* $\langle v, v'\rangle > 0$ *and consider a vector* $u = u_t$ *such that* $\|u_V\| \geq c > 0$. *Then, there exists* $T := T(c, \theta, \eta)$ *such that for* $u' := u_{t+T}$, *with probability at least* 2^{-T}, *vector* u'_V *will either be a conical combination of* v *and* v' or *a conical combination of* $-v$ *and* $-v'$.

Proof. First, for any vector u such that $\|u_V\| \geq c > 0$, it is clear that at least one of $v, v', -v, -v'$ has positive inner product with u (and u_V) which can be lower bounded by a function of c, θ, and η (see Figure 5-6). Take such a vector and call it v^*. Applying it repeatedly will make u'_V arbitrarily close to v^* (cf. Proposition 5.9.1).

Finally, after choosing the number of applications of v^* such that both $\|u' - u'_V\|$ and $\|u'_V - v^*\|$ are small enough, we apply the other intervention vector (v or v') once. It is clear that at this stage vector u'_V either already is in the convex cone (and the additional intervention keeps it inside) or the intervention with the other vector brings it inside the cone.

Therefore, there exists a sequence of $T(c,\theta,\eta)$ interventions that make u_V enter cone(v,v') or the cone$(-v,-v')$. $\square$

We combine Propositions 5.9.1, 5.9.2 and 5.9.3 to show that when $\langle v,v'\rangle \neq 0$, almost any vector u eventually approaches one of the convex cones (cone(v,v') or cone$(-v,-v')$) as time goes to infinity.

Proof of Theorem 5.4.15. Let $\|u_V\| = c > 0$. Proposition 5.9.1 tells us that the squared norm of the component u_W in the subspace $W = V^\perp$ never increases, and with probability 1/2 decreases by a multiplicative factor $(1-(\eta^2/2+\eta)\cdot c^2\theta^2/16)$. By induction (note that c only increases with successive applications), u_W converges to 0, and consequently $\|u-u_V\|$ converges to 0, almost surely.

In order to show that additionally convergence to one of the two convex cones occurs, we apply Proposition 5.9.3. Since at *any time step t*, there exists a sequence of T choices that puts u_V in one of the convex cones, and since T depends only on the starting parameters c, θ, and η, we get that u_V almost surely eventually enters one of the cones. By Proposition 5.9.2 and induction, once u_V enters a convex cone, it never leaves. $\square$

As a corollary of Propositions 5.9.1 and 5.9.2, when $\langle v,v'\rangle = 0$, u_V always stays in the quadrant it starts in.

Proof of Corollary 5.4.17. The first statement is an inductive application of Proposition 5.9.1, exactly the same as in the proof of Theorem 5.4.15.

The second statement follows from noting that out of four orthogonal pairs of vectors $\{v,v'\}$, $\{v,-v'\}$, $\{-v,v'\}$, or $\{-v,-v'\}$, there is exactly one such that u_V is a (strict) conical combination of this pair (by assuming $\langle u,v\rangle \neq 0$ and $\langle u,v'\rangle \neq 0$ we avoid ambiguity in case u_V is parallel to v or v'). By the same argument as in Proposition 5.9.2 and by induction, if the initial projection u_V is strictly inside one of the convex cones, it remains strictly inside forever. $\square$

5.9.2 Proof of Theorem 5.4.16

Consider the subspace V with some coordinate system (cf. Figure 5-6) imposed on it. As is standard, a unit vector $u \in V$ can be represented in this system by its primary angle $\alpha(u) \in [0, 2\pi)$ as measured clockwise from the positive x-axis.

Given a unit vector $v^* \in V$, let $f_{v^*} : [0, 2\pi) \to [0, 2\pi)$ be the function with the following meaning: Given a unit vector $u \in V$ with angle $\alpha = \alpha(u)$, the value $f_{v^*}(\alpha) = \alpha(u')$ represents the angle of vector u' resulting from applying intervention v^* to vector u. Note that $\alpha(v^*)$ is a fixed point of f_{v^*}. Also, the functions f_v and $f_{v'}$ map the the cone(v, v') to itself.

The main part of our argument is the following lemma, which we prove last:

Lemma 5.9.4. *Functions f_v and $f_{v'}$ restricted to the convex cone of v and v' are contractions, i.e., there exists $k = k(\theta, \eta) < 1$ such that for all vectors $u, u' \in \mathrm{cone}(v, v')$, letting $\alpha := \alpha(u), \beta := \alpha(u'), v^* \in \{v, v'\}$, we have*

$$\left| f_{v^*}(\beta) - f_{v^*}(\alpha) \right| \leq k \cdot |\beta - \alpha| \,, \tag{5.24}$$

where the distances $|f_{v^*}(\beta) - f_{v^*}(\alpha)|$ and $|\beta - \alpha|$ are in the metric induced by S^1, i.e., "modulo 2π".

Lemma 5.9.4 implies that the angle distance between two opinions $u_1^{(t)}, u_2^{(t)} \in V$ starting in the convex cone (deterministically) converges to 0 as t goes to infinity. Of course, this is equivalent to their Euclidean distance $\|u_1^{(t)} - u_2^{(t)}\|$ converging to 0. We now make a continuity argument to show that convergence almost surely occurs also for $u_1^{(t)}, u_2^{(t)} \notin V$. To this end, we define $g_v, g_{v'} : S^{d-1} \to [0, 2\pi)$ as natural extensions of $f_v, f_{v'}$: the value $g_{v^*}(u)$ denotes the angle of the projection u'_V of the new opinion onto V, after applying v^* on opinion u (cf. Figure 5-6). Note that the value $g_{v^*}(u)$ depends only on the angle $\alpha(u_V)$ and the projection length $\|u_W\|$:

$$g_{v^*}(u) = g_{v^*}\Big(\alpha(u_V), \|u_W\|\Big) \,.$$

In this parametrization, for $u \in V$ we have $f_{v^*}(\alpha(u)) = g_{v^*}(u) = g_{v^*}(\alpha(u), 0)$.

By Theorem 5.4.15, for any starting opinions $u_1^{(1)}$ and $u_2^{(1)}$ satisfying the assumptions, almost surely there exists a t such that $(u_1^{(t)})_V$ and $(u_2^{(t)})_V$ end up inside (possibly different) convex cones. We consider the case of $u_1^{(t)}$ and $u_2^{(t)}$ both in cone(v, v'), other three cases being analogous. Furthermore, almost surely, $\|(u_1^{(t)})_W\|$ and $\|(u_2^{(t)})_W\|$ converge to 0. Hence, it is enough that we show that almost surely $|\alpha((u_1^{(t)})_V) - \alpha((u_2^{(t)})_V)|$ (again using S^1 distance) converges to zero.

To this end, let $\delta > 0$. By uniform continuity of f_v, we know that for small enough value of $r = \|u_W\|$, we have

$$|f_v(\alpha, r) - f_v(\alpha, 0)| < \frac{1-k}{4} \cdot \delta$$

for every $\alpha \in [0, 2\pi)$, where k is the Lipschitz constant from (5.24). Therefore, almost surely, for t large enough, for $u_1^{(t)}$ and $u_2^{(t)}$ parameterized as $u_1^{(t)} = (\alpha_1, r_1)$ and $u_2^{(t)} = (\alpha_2, r_2)$ we have

$$\begin{aligned} |f_v(\alpha_1, r_1) - f_v(\alpha_2, r_2)| &\le |f_v(\alpha_1, r_1) - f_v(\alpha_1, 0)| + |f_v(\alpha_1, 0) - f_v(\alpha_2, 0)| + |f_v(\alpha_2, 0) - f_v(\alpha_2, r_2)| \\ &\le \frac{1-k}{4} \cdot \delta + k \cdot |\alpha_1 - \alpha_2| + \frac{1-k}{4} \cdot \delta \le \left(k + \frac{1-k}{2}\right) \cdot \max(|\alpha_1 - \alpha_2|, \delta) \, . \end{aligned}$$

Since $k + (1-k)/2 < 1$, and applying the same argument to $f_{v'}$, we conclude by induction that the distance $|\alpha_1(t) - \alpha_2(t)|$ must go and stay below δ in a finite number of steps. Since $\delta > 0$ was arbitrary, it must be that $|\alpha_1(t) - \alpha_2(t)|$ converges to 0, concluding the proof of Theorem 5.4.16. □

It remains to prove Lemma 5.9.4:

Proof of Lemma 5.9.4. Let $f := f_{(1,0)}$, i.e., f corresponds to the intervention along the x-axis. Clearly, functions f_v and $f_{v'}$ are cyclic shifts of f. More precisely, we have

$$f_{v*}(\alpha) = \alpha(v^*) + f\Big(\alpha - \alpha(v^*)\Big) \,, \tag{5.25}$$

where arithmetic in (5.25) is modulo 2π. Furthermore, f is symmetric around the intervention vector, i.e., $f(\alpha) = 2\pi - f(2\pi - \alpha)$ for $0 \leq \alpha \leq \pi$. Hence, to prove that f_v and $f_{v'}$ restricted to cone(v, v') are contractions, it is enough that we show that f restricted to the interval $[0, \theta]$ is a contraction (recall that we assumed $\cos^2(\theta) > 1/(2+\eta)$).

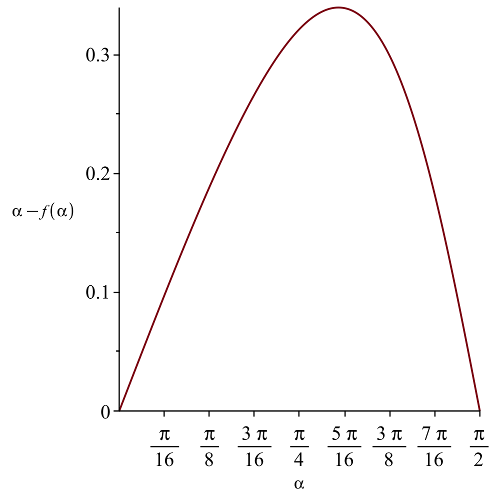


Figure 5-7: *The graph of the "pull function" $\alpha - f(\alpha)$ in case $\eta = 1$.*

To that end, a computation gives the formula for f

$$f(\alpha) = \arccos\left(\frac{(1+\eta)\cos\alpha}{\sqrt{1+(\eta^2+2\eta)\cos^2\alpha}}\right). \tag{5.26}$$

More computation establishes that, additionally, for every $0 \leq \alpha < \beta \leq \pi/2$:

1. $f(\alpha) \leq \alpha$. In other words, applying the intervention brings vector u closer to the intervention vector.

2. $f(\alpha) < f(\beta)$, i.e., applying the intervention does not change relative ordering of vectors wrt the intervention vector.

3. If $\beta \leq \theta^* := \arccos\left(\sqrt{\frac{1}{2+\eta}}\right)$, then $\alpha - f(\alpha) < \beta - f(\beta)$, i.e., in absolute terms, the "pull" on a vector is stronger the further away it is from the intervention vector (until the correlation reaches the threshold $1/\sqrt{2+\eta}$, cf. Figure 5-7).

The preceding items taken together imply that for every $0 \leq \alpha < \beta \leq \theta^*$ we have $0 <$

$f(\beta) - f(\alpha) < \beta - \alpha$. To conclude that f is a contraction, we observe that for any $\theta < \theta^*$ we have that f and f' are continuous on the interval $[0, \theta]$ and that $f'(\alpha) > 0$ for $0 \leq \alpha \leq \theta$. $\square$

Bibliography

[1] 23andMe. About us. https://mediacenter.23andme.com/company/about-us/.

[2] Abhinav Aggarwal, Varsha Dani, Thomas P. Hayes, and Jared Saia. Distributed computing with channel noise. Cryptology ePrint Archive, Report 2017/710, 2017. https://eprint.iacr.org/2017/710.

[3] Shweta Agrawal, Ran Gelles, and Amit Sahai. Adaptive protocols for interactive communication. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 595–599, 2016.

[4] Noga Alon, Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Reliable communication over highly connected noisy networks. In *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing*, PODC '16, pages 165–173, 2016.

[5] Noga Alon, Oded Goldreich, Johan Håstad, and René Peralta. Simple constructions of almost k-wise independent random variables. *Random Structures & Algorithms*, 3(3):289–304, 1992.

[6] Noga Alon, Elchanan Mossel, and Robin Pemantle. Distributed corruption detection in networks. *Theory of Computing*, 16(1):1–23, 2020.

[7] Edoardo Amaldi and Viggo Kann. On the approximability of minimizing nonzero variables or unsatisfied relations in linear systems. *Theoretical Computer Science*, 209(1–2):237–260, 1998.

[8] Ancestry.com. Ancestry continues to lead the industry with world's largest consumer dna network. https://www.ancestry.com/corporate/newsroom/press-releases/ancestry%C2%AE-surpasses-15-million-members-its-dna-network-powering-unparalleled.

[9] Per Austrin, Toniann Pitassi, and Yu Wu. Inapproximability of treewidth, one-shot pebbling, and related layout problems. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques - 15th International Workshop, APPROX 2012, and 16th International Workshop, RANDOM 2012, Cambridge, MA, USA, August 15-17, 2012. Proceedings*, pages 13–24, 2012.

[10] Robert Axelrod. The dissemination of culture: A model with local convergence and global polarization. *Journal of Conflict Resolution*, 41(2):203–226, 1997.

[11] Christopher A. Bail, Lisa P. Argyle, Taylor W. Brown, John P. Bumpus, Haohan Chen, M. B. Fallin Hunzaker, Jaemin Lee, Marcus Mann, Friedolin Merhout, and Alexander Volfovsky. Exposure to opposing views on social media can increase political polarization. *Proceedings of the National Academy of Sciences*, 115(37):9216–9221, 2018.

[12] Eytan Bakshy, Solomon Messing, and Lada A. Adamic. Exposure to ideologically diverse news and opinion on Facebook. *Science*, 348(6239):1130–1132, 2015.

[13] Venkatesh Bala and Sanjeev Goyal. Learning from neighbours. *The Review of Economic Studies*, 65(3):595–621, 1998.

[14] Delia Baldassarri. *Crosscutting Social Spheres? Political Polarization and the Social Roots of Pluralism*. PhD thesis, Columbia University, 2007.

[15] Delia Baldassarri and Peter Bearman. Dynamics of political polarization. *American Sociological Review*, 72(5):784–811, 2007.

[16] Delia Baldassarri and Andrew Gelman. Partisans without constraint: Political polarization and trends in American public opinion. *American Journal of Sociology*, 114(2):408–446, 2008.

[17] Walid Ben-Ameur, Mohamed-Ahmed Mohamed-Sidi, and José Neto. The k - separator problem: polyhedra, complexity and approximation results. *J. Comb. Optim.*, 29(1):276–307, 2015.

[18] Shai Ben-David, Nadav Eiron, and Philip M. Long. On the difficulty of approximately maximizing agreements. *Journal of Computer and System Sciences*, 66(3):496–514, 2003.

[19] Shai Ben-David, Nadav Eiron, and Hans Ulrich Simon. The computational complexity of densest region detection. *Journal of Computer and System Sciences*, 64(1):22–47, 2002.

[20] Shai Ben-David and Hans-Ulrich Simon. Efficient learning of linear perceptrons. In *Advances in Neural Information Processing Systems (NIPS)*, pages 189–195, 2000.

[21] David Blackwell, Leo Breiman, and A. J. Thomasian. The capacities of certain channel classes under random coding. *The Annals of Mathematical Statistics*, 31(3):558–567, 1960.

[22] Richard A Blythe and Alan J McKane. Stochastic models of evolution in genetics, ecology and linguistics. *Journal of Statistical Mechanics: Theory and Experiment*, 2007(07):P07018, 2007.

[23] Hans L. Bodlaender, John R. Gilbert, Hjálmtyr Hafsteinsson, and Ton Kloks. Approximating treewidth, pathwidth, frontsize, and shortest elimination tree. *J. Algorithms*, 18(2):238–255, 1995.

[24] Zvika Brakerski, Yael T. Kalai, and Moni Naor. Fast interactive coding against adversarial noise. *J. ACM*, 61(6):35:1–35:30, December 2014.

[25] Zvika Brakerski and Yael Tauman Kalai. Efficient interactive coding against adversarial noise. *Proceedings of the IEEE Symposium on Foundations of Computer Science*, pages 160–166, 2012.

[26] M. Braverman, R. Gelles, J. Mao, and R. Ostrovsky. Coding for interactive communication correcting insertions and deletions. *IEEE Transactions on Information Theory*, 63(10):6256–6270, Oct 2017.

[27] M. Braverman and A. Rao. Toward coding for maximum errors in interactive communication. *Information Theory, IEEE Transactions on*, 60(11):7248–7255, Nov 2014.

[28] Mark Braverman and Klim Efremenko. List and unique coding for interactive communication in the presence of adversarial noise. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 236–245, 2014.

[29] Mark Braverman, Klim Efremenko, Ran Gelles, and Bernhard Haeupler. Constant-rate coding for multiparty interactive communication is impossible. *J. ACM*, 65(1):4:1–4:41, December 2018.

[30] Nader H. Bshouty and Lynn Burroughs. Maximizing agreements and coagnostic learning. *Theoretical Computer Science*, 350(1):24–39, 2006.

[31] Keren Censor-Hillel, Ran Gelles, and Bernhard Haeupler. Making Asynchronous Distributed Computations Robust to Channel Noise. In Anna R. Karlin, editor, *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*, volume 94 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:20, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[32] Keren Censor-Hillel, Ran Gelles, and Bernhard Haeupler. Making asynchronous distributed computations robust to noise. *Distributed Computing*, 32(5):405–421, Oct 2019.

[33] Pew Researcher Center. Political polarization in the American public: How increasing ideological uniformity and partisan antipathy affect politics, compromise and everyday life, 2014.

[34] Gil Cohen, Bernhard Haeupler, and Leonard J Schulman. Explicit binary tree codes with polylogarithmic size alphabet. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 535–544, 2018.

[35] Michael D. Conover, Jacob Ratkiewicz, Matthew Francisco, Bruno Gonçalves, Filippo Menczer, and Alessandro Flammini. Political polarization on Twitter. In *International Conference on Weblogs and Social Media (ICWSM)*, pages 89–96, 2011.

[36] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, pages 471–488, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[37] Anton T. Dahbura and Gerald M. Masson. An $O(n^{2.5})$ fault identification algorithm for diagnosable systems. *IEEE Trans. Computers*, 33(6):486–492, 1984.

[38] Pranav Dandekar, Ashish Goel, and David T. Lee. Biased assimilation, homophily, and the dynamics of polarization. *Proceedings of the National Academy of Sciences*, 110(15):5791–5796, 2013.

[39] Constantinos Daskalakis, Elchanan Mossel, and Sébastien Roch. Optimal phylogenetic reconstruction. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pages 159–168, 2006.

[40] M. C. Davey and D. J. C. Mackay. Reliable communication over channels with insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 47(2):687–698, Feb 2001.

[41] Michela Del Vicario, Antonio Scala, Guido Caldarelli, H. Eugene Stanley, and Walter Quattrociocchi. Modeling confirmation bias and polarization. *Scientific Reports*, 7:40391, 2017.

[42] Irit Dinur, Elchanan Mossel, and Oded Regev. Conditional hardness for approximate coloring. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 344–353, New York, NY, USA, 2006. ACM.

[43] Klim Efremenko, Elad Haramaty, and Yael Kalai. Interactive coding with constant round and communication blowup. *Electronic Colloquium on Computational Complexity (ECCC)*, 25:54, 2018.

[44] Péter L Erdős, Michael A Steel, László A Székely, and Tandy J Warnow. A few logs suffice to build (almost) all trees (i). *Random Structures & Algorithms*, 14(2):153–184, 1999.

[45] Yaniv Erlich, Tal Shor, Itsik Pe'er, and Shai Carmi. Identity inference of genomic data using long-range familial searches. *Science*, 362(6415):690–694, 2018.

[46] William Evans, Claire Kenyon, Yuval Peres, and Leonard J. Schulman. Broadcasting on trees and the ising model. *Ann. Appl. Probab.*, 10(2):410–433, 05 2000.

[47] Vitaly Feldman, Parikshit Gopalan, Subhash Khot, and Ashok Kumar Ponnuswami. New results for learning noisy parities and halfspaces. In *Symposium on Foundations of Computer Science (FOCS)*, pages 563–574, 2006.

[48] Morris P. Fiorina and Samuel J. Abrams. Political polarization in the American public. *Annual Reviev of Political Science*, 11:563–588, 2008.

[49] Morris P. Fiorina, Samuel J. Abrams, and Jeremy C. Pope. *Culture War? The Myth of a Polarized America*. Pearson-Longman, 2005.

[50] Odd-Helge Fjeldstad. Fighting fiscal corruption: lessons from the tanzania revenue authority. *Public Administration and Development: The International Journal of Management Research and Practice*, 23(2):165–175, 2003.

[51] Jacey Fortin. A list of the companies cutting ties with the N.R.A. The New York Times website, 24 February 2018. `https://www.nytimes.com/2018/02/24/business/nra-companies-boycott.html`, 2018.

[52] Kiran Garimella. *Polarization on Social Media*. PhD thesis, Aalto University, 2018. 20/2018.

[53] Kiran Garimella, Aristides Gionis, Nikos Parotsidis, and Nikolaj Tatti. Balancing information exposure in social networks. In *Advances in Neural Information Processing Systems (NIPS)*, pages 4663–4671, 2017.

[54] Kiran Garimella, Gianmarco De Francisci Morales, Aristides Gionis, and Michael Mathioudakis. Reducing controversy by connecting opposing views. In *International Conference on Web Search and Data Mining (WSDM)*, pages 81–90. ACM, 2017.

[55] R. Gelles, B. Haeupler, G. Kol, N. Ron-Zewi, and A. Wigderson. Explicit capacity approaching coding for interactive communication. *IEEE Transactions on Information Theory*, 64(10):6546–6560, Oct 2018.

[56] Ran Gelles. Coding for interactive communication: A survey. *Foundations and Trends® in Theoretical Computer Science*, 13(1–2):1–157, 2017.

[57] Ran Gelles and Yael T. Kalai. Constant-Rate Interactive Coding Is Impossible, Even In Constant-Degree Networks. In Christos H. Papadimitriou, editor, *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, volume 67 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:13, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[58] Ran Gelles, Yael T. Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding, part I: Oblivious insertions, deletions and substitutions. submitted, 2019.

[59] Ran Gelles, Yael T. Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding, part II: Non-oblivious noise. submitted, 2019.

[60] Ran Gelles, Yael Tauman Kalai, and Govind Ramnarayan. Efficient multiparty interactive coding for insertions, deletions, and substitutions. In *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing*, PODC '19, pages 137–146, New York, NY, USA, 2019. ACM.

[61] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient and explicit coding for interactive communication. In *Proceeding of the IEEE Symposium on Foundations of Computer Science*, FOCS '11, pages 768–777, 2011.

[62] Ran Gelles, Ankur Moitra, and Amit Sahai. Efficient coding for interactive communication. *Information Theory, IEEE Transactions on*, 60(3):1899–1913, March 2014.

[63] Daniel Genkin, Yuval Ishai, Manoj M. Prabhakaran, Amit Sahai, and Eran Tromer. Circuits resilient to additive attacks with applications to secure computation. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, STOC '14, pages 495–504, 2014.

[64] Mohsen Ghaffari and Bernhard Haeupler. Optimal Error Rates for Interactive Coding II: Efficiency and List Decoding. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 394–403, 2014.

[65] Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM Journal on computing*, 18(1):186–208, 1989.

[66] V. Guruswami and P. Indyk. Linear-time encodable/decodable codes with near-optimal rate. *IEEE Trans. on Information Theory*, 51(10):3393–3400, 2005.

[67] V. Guruswami and R. Li. Efficiently decodable insertion/deletion codes for high-noise and high-rate regimes. In *2016 IEEE International Symposium on Information Theory (ISIT)*, pages 620–624, July 2016.

[68] Venkatesan Guruswami and Prasad Raghavendra. Hardness of learning halfspaces with noise. *SIAM Journal on Computing*, 39(2):742–765, 2009.

[69] Venkatesan Guruswami and Adam Smith. Codes for computationally simple channels: Explicit constructions with optimal rate. In *FOCS '10*, pages 723–732, 2010.

[70] Bernhard Haeupler. Interactive Channel Capacity Revisited. In *Proceedings of the IEEE Symposium on Foundations of Computer Science*, FOCS '14, pages 226–235, 2014.

[71] Bernhard Haeupler, Amirbehshad Shahrasbi, and Ellen Vitercik. Synchronization strings: Channel simulations and interactive coding for insertions and deletions. *CoRR*, abs/1707.04233, 2017.

[72] S. Louis Hakimi and A. T. Amin. Characterization of connection assignment of diagnosable systems. *IEEE Trans. Computers*, 23(1):86–88, 1974.

[73] Richard W. Hamming. Error detecting and error correcting codes. *Bell System technical journal*, 29(2):147–160, 1950.

[74] Jan Hązła, Yan Jin, Elchanan Mossel, and Govind Ramnarayan. A geometric model of opinion polarization. *arXiv preprint arXiv:1910.05274*, 2019.

[75] Dan He, Zhanyong Wang, Buhm Han, Laxmi Parida, and Eleazar Eskin. Iped: inheritance path-based pedigree reconstruction algorithm using genotype data. *Journal of Computational Biology*, 20(10):780–791, 2013.

[76] Dan He, Zhanyong Wang, Laxmi Parida, and Eleazar Eskin. Iped2: Inheritance path based pedigree reconstruction algorithm for complicated pedigrees. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, BCB '14, page 202–210, New York, NY, USA, 2014. Association for Computing Machinery.

[77] Rainer Hegselmann and Ulrich Krause. Opinion dynamics and bounded confidence. models, analysis, and simulation. *Journal of Artificial Societies and Social Simulation*, 5(3), 2002.

[78] William M. Hoza, 2015. Private communication.

[79] William M. Hoza and Leonard J. Schulman. The adversarial noise threshold for distributed protocols. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 240–258, 2016.

[80] Daniel Hsu, Sham M Kakade, and Tong Zhang. A spectral algorithm for learning hidden markov models. *Journal of Computer and System Sciences*, 78(5):1460–1480, 2012.

[81] Jisca Huisman. Pedigree reconstruction from snp data: parentage assignment, sibship clustering and beyond. *Molecular ecology resources*, 17(5):1009–1024, 2017.

[82] National Human Genome Research Institute. The cost of sequencing a human genome. https://www.genome.gov/about-genomics/fact-sheets/Sequencing-Human-Genome-cost, 2019.

[83] Shanto Iyengar and Sean J. Westwood. Fear and loathing across party lines: New evidence on group polarization. *American Journal of Political Science*, 59(3):690–707, 2015.

[84] A. Jadbabaie, N. Motee, and M. Barahona. On the stability of the Kuramoto model of coupled nonlinear oscillators. In *Proceedings of the 2004 American Control Conference*, volume 5, pages 4296–4301 vol.5, June 2004.

[85] Abhishek Jain, Yael Tauman Kalai, and Allison Lewko. Interactive coding for multiparty protocols. In *Proceedings of the 6th Conference on Innovations in Theoretical Computer Science, ITCS '15*, pages 1–10, 2015.

[86] Yan Jin, Elchanan Mossel, and Govind Ramnarayan. Being corrupt requires being clever, but detecting corruption doesn't. In *10th Innovations in Theoretical Computer Science Conference (ITCS 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2019.

[87] Tiko Kameda, S Toida, and FJ Allan. A diagnosing algorithm for networks. *Information and Control*, 29(2):141–148, 1975.

[88] David Kempe, Jon Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 137–146. ACM, 2003.

[89] David Kempe, Jon M. Kleinberg, and Éva Tardos. Maximizing the spread of influence through a social network. *Theory of Computing*, 11:105–147, 2015.

[90] Subhash Khot. On the power of unique 2-prover 1-round games. In *Proceedings of the 17th Annual IEEE Conference on Computational Complexity, Montréal, Québec, Canada, May 21-24, 2002*, page 25, 2002.

[91] Junhyong Kim, Elchanan Mossel, Miklós Z Rácz, and Nathan Ross. Can one hear the shape of a population history? *Theoretical population biology*, 100:26–38, 2015.

[92] Younhun Kim, Frederic Koehler, Ankur Moitra, Elchanan Mossel, and Govind Ramnarayan. How many subpopulations is too many? exponential lower bounds for inferring population histories. *Journal of Computational Biology*, 2019.

[93] Younhun Kim, Elchanan Mossel, Govind Ramnarayan, and Paxton Turner. Efficient reconstruction of stochastic pedigrees. *arXiv preprint arXiv:2005.03810*, 2020.

[94] Marek Kimmel and David Axelrod. *Branching Processes in Biology*. Springer Publishing Company, Incorporated, 2nd edition, 2015.

[95] Bonnie Kirkpatrick, Shuai Cheng Li, Richard M Karp, and Eran Halperin. Pedigree reconstruction using identity by descent. *Journal of Computational Biology*, 18(11):1481–1493, 2011.

[96] Gillat Kol and Ran Raz. Interactive channel capacity. In *STOC '13: Proceedings of the 45th annual ACM Symposium on theory of computing*, pages 715–724, 2013.

[97] Gina Kolata and Heather Murphy. The golden state killer is tracked through a thicket of dna, and experts shudder. *The New York Times*, 4 2018.

[98] Stefan Krasa and Mattias K. Polborn. Political competition in legislative elections. *American Political Science Review*, 112(4):809–825, 2018.

[99] Jon G Kuhl and Sudhakar M Reddy. Distributed fault-tolerance for large multiprocessor systems. In *Proceedings of the 7th annual symposium on Computer Architecture*, pages 23–30. ACM, 1980.

[100] Ilyana Kuziemko and Ebonya Washington. Why did the democrats lose the south? bringing new data to an old debate. *American Economic Review*, 108(10):2830–67, 2018.

[101] Steven Lalley. Poisson processes. *Statistics 312: Stochastic Processes*, 2016.

[102] Leslie Lamport. The part-time parliament. In *Concurrency: the Works of Leslie Lamport*, pages 277–317. 2019.

[103] Leslie Lamport et al. Paxos made simple. *ACM Sigact News*, 32(4):18–25, 2001.

[104] Howard Lavine, Eugene Borgida, and John L. Sullivan. On the relationship between attitude involvement and attitude accessibility: Toward a cognitive-motivational model of political information processing. *Political Psychology*, 21(1):81–106, 2000.

[105] Euiwoong Lee. Partitioning a graph into small pieces with applications to path transversal. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 1546–1558, 2017.

[106] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[107] V. I. Levenshtein. On perfect codes in deletion and insertion metric. *Discrete Mathematics and Applications*, 2:241—258, 1992.

[108] Charles G. Lord, Lee Ross, and Mark R. Lepper. Biased assimilation and attitude polarization: The effects of prior theories on subsequently considered evidence. *Journal of Personality and Social Psychology*, 37(11):2098–2109, 1979.

[109] Nancy A Lynch. *Distributed algorithms*. Elsevier, 1996.

[110] Michael W. Macy, James A. Kitts, Andreas Flache, and Steve Benard. Polarization in dynamic networks: A Hopfield model of emergent structure. *Dynamic social network modeling and analysis*, pages 162–173, 2003.

[111] Shachindra N. Maheshwari and S. Louis Hakimi. On models for diagnosable systems and probabilistic fault diagnosis. *IEEE Trans. Computers*, 25(3):228–236, 1976.

[112] Anuran Makur, Elchanan Mossel, and Yury Polyanskiy. Broadcasting on bounded degree dags. *arXiv preprint arXiv:1803.07527*, 2018.

[113] Elchanan Mossel. Reconstruction on trees: beating the second eigenvalue. *Annals of Applied Probability*, pages 285–300, 2001.

[114] Elchanan Mossel. On the impossibility of reconstructing ancestral data and phylogenies. *Journal of computational biology*, 10(5):669–676, 2003.

[115] Elchanan Mossel. Phase transitions in phylogeny. *Transactions of the American Mathematical Society*, 356(6):2379–2404, 2004.

[116] Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375, 2005.

[117] Elchanan Mossel and Sébastien Roch. Learning nonsingular phylogenies and hidden markov models. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 366–375, 2005.

[118] Elchanan Mossel and Grant Schoenebeck. Reaching consensus on social networks. *Innovations in Computer Science*, 2010.

[119] Thebeth Rufaro Mukwembi and Simon Mukwembi. Corruption and its detection: a graph-theoretic approach. *Computational and Mathematical Organization Theory*, 23(2):293–300, Jun 2017.

[120] Sendhil Mullainathan and Andrei Shleifer. The market for news. *American Economic Review*, 95(4):1031–1053, 2005.

[121] MyHeritage. Myheritage end-of-year infographic. `https://blog.myheritage.com/2019/12/wrapping-up-a-fantastic-2019`.

[122] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput*, 22:838–856, 1993.

[123] Anand Kumar Narayanan and Matthew Weidner. On decoding cohen-haeupler-schulman tree codes. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1337–1356. SIAM, 2020.

[124] Giang-Truong Nguyen and Kyungbaek Kim. A survey about consensus algorithms used in blockchain. *Journal of Information processing systems*, 14(1), 2018.

[125] Richard P Nielsen. Corruption networks and implications for ethical corruption reform. *Journal of Business ethics*, 42(2):125–149, 2003.

[126] Mark Noah. Beyond individual differences: Social differentiation from first principles. *American Sociological Review*, 63(3):309, 1998.

[127] Andrzej Nowak, Jacek Szamrej, and Bibb Latané. From private attitude to public opinion: A dynamic theory of social impact. *Psychological Review*, 97(3):362, 1990.

[128] Maarten Oosten, Jeroen HGC Rutten, and Frits CR Spieksma. Disconnecting graphs by removing vertices: a polyhedral approach. *Statistica Neerlandica*, 61(1):35–60, 2007.

[129] Eli Pariser. *The Filter Bubble: How the New Personalized Web Is Changing What We Read and How We Think*. Penguin, New York, 2011.

[130] Sergey E. Parsegov, Anton V. Proskurnikov, Roberto Tempo, and Noah E. Friedkin. Novel multidimensional models of opinion dynamics in social networks. *IEEE Transactions on Automatic Control*, 62(5):2270–2285, May 2017.

[131] Marshall Pease, Robert Shostak, and Leslie Lamport. Reaching agreement in the presence of faults. *Journal of the ACM (JACM)*, 27(2):228–234, 1980.

[132] Jacopo Perego and Sevgi Yuksel. Media competition and social disagreement. Working paper, 2018.

[133] Thomas F. Pettigrew and Linda R. Tropp. A meta-analytic test of intergroup contact theory. *Journal of Personality and Social Psychology*, 90(5):751–783, 2006.

[134] D Pollard. Mini empirical. Manuscript. http://www.stat.yale.edu/pollard/Books/Mini, 2015.

[135] Franco P. Preparata, Gernot Metze, and Robert T. Chien. On the connection assignment problem of diagnosable systems. *IEEE Trans. Electronic Computers*, 16(6):848–854, 1967.

[136] Markus Prior. Media and political polarization. *Annual Review of Political Science*, 16:101–127, 2013.

[137] Prasad Raghavendra and David Steurer. Graph expansion and the unique games conjecture. In *Proceedings of the 42nd ACM Symposium on Theory of Computing, STOC 2010, Cambridge, Massachusetts, USA, 5-8 June 2010*, pages 755–764, 2010.

[138] Prasad Raghavendra, David Steurer, and Madhur Tulsiani. Reductions between expansion problems. In *Proceedings of the 27th Conference on Computational Complexity, CCC 2012, Porto, Portugal, June 26-29, 2012*, pages 64–73, 2012.

[139] Sridhar Rajagopalan and Leonard Schulman. A coding theorem for distributed computation. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 790–799, 1994.

[140] Kazutoshi Sasahara, Wen Chen, Hao Peng, Giovanni Luca Ciampaglia, Alessandro Flammini, and Filippo Menczer. On the inevitability of online echo chambers. arXiv:1905.03919, 2019.

[141] Elizabeth A. Saylor, Katherine A. Vittes, and Susan B. Sorenson. Firearm advertising: Product depiction in consumer gun magazines. *Evaluation Review*, 28(5):420–433, 2004.

[142] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi. Codes correcting a burst of deletions or insertions. *IEEE Transactions on Information Theory*, 63(4):1971–1985, April 2017.

[143] L. J. Schulman and D. Zuckerman. Asymptotically good codes correcting insertions, deletions, and transpositions. *IEEE Transactions on Information Theory*, 45(7):2552–2557, Nov 1999.

[144] Leonard J. Schulman. Communication on noisy channels: a coding theorem for computation. *Foundations of Computer Science, Annual IEEE Symposium on*, pages 724–733, 1992.

[145] Leonard J. Schulman. Coding for interactive communication. *IEEE Transactions on Information Theory*, 42(6):1745–1756, 1996.

[146] F. Sellers. Bit loss and gain correction code. *IRE Transactions on Information Theory*, 8(1):35–38, January 1962.

[147] Charles Semple and Mike Steel. Phylogenetics. 24, 2003.

[148] Claude E. Shannon. A mathematical theory of communication. *ACM SIGMOBILE Mobile Computing and Communications Review*, 5(1):3–55, 2001. Originally appeared in *Bell System Tech. J.* 27:379–423, 623–656, 1948.

[149] Doron Shem-Tov and Eran Halperin. Historical pedigree reconstruction from extant populations using partitioning of relatives (prepare). *PLoS computational biology*, 10(6), 2014.

[150] A. A. Sherstov and P. Wu. Optimal interactive coding for insertions, deletions, and substitutions. *IEEE Transactions on Information Theory*, 65(10):5971–6000, Oct 2019.

[151] Guodong Shi, Alexandre Proutiere, Mikael Johansson, John S. Baras, and Karl H. Johansson. The evolution of beliefs over signed social networks. *Operations Research*, 64(3):585–604, 2016.

[152] John Sides and Daniel J. Hopkins. *Political polarization in American politics*. Bloomsbury Publishing USA, 2015.

[153] Brendan Snyder. LGBT advertising: How brands are taking a stance on issues. Think with Google, 2015.

[154] Mike Steel and Jotun Hein. Reconstructing pedigrees: a combinatorial perspective. *Journal of theoretical biology*, 240(3):360–367, 2006.

[155] Madhu Sudan. Coding theory: Tutorial & survey. In *Proceedings 42nd IEEE Symposium on Foundations of Computer Science*, pages 36–53. IEEE, 2001.

[156] Gregory F. Sullivan. A polynomial time algorithm for fault diagnosability. In *FOCS*, pages 148–156. IEEE Computer Society, 1984.

[157] E. Tanaka and T. Kasai. Synchronization and substitution error-correcting codes for the levenshtein metric. *IEEE Transactions on Information Theory*, 22(2):156–162, March 1976.

[158] G. Tenengolts. Nonbinary codes, correcting single deletion or insertion (corresp.). *IEEE Transactions on Information Theory*, 30(5):766–769, Sep. 1984.

[159] Bhalchandra D Thatte and Mike Steel. Reconstructing pedigrees: a stochastic perspective. *Journal of theoretical biology*, 251(3):440–449, 2008.

[160] Elizabeth A. Thompson. Statistical inference from genetic data on pedigrees. *NSF-CBMS Regional Conference Series in Probability and Statistics*, 6:i–169, 2000.

[161] Elizabeth A Thompson. Identity by descent: variation in meiosis, across genomes, and in populations. *Genetics*, 194(2):301–326, 2013.

[162] J. Ullman. On the capabilities of codes to correct synchronization errors. *IEEE Transactions on Information Theory*, 13(1):95–105, January 1967.

[163] Donald E. Vinson, Jerome E. Scott, and Lawrence M. Lamont. The role of personal values in marketing and consumer behavior. *Journal of Marketing*, 41(2):44–50, 1977.

[164] Jinliang Wang. Pedigree reconstruction from poor quality genotype data. *Heredity*, 122(6):719–728, 2019.

[165] Hywel T.P. Williams, James R. McMurray, Tim Kurz, and F. Hugo Lambert. Network analysis reveals open forums and echo chambers in social media discussions of climate change. *Global Environmental Change*, 32:126–138, 2015.

[166] Jie Xu and Shi-ze Huang. Sequentially t-diagnosable systems: A characterization and its applications. *IEEE Trans. Computers*, 44(2):340–345, 1995.

[167] Sergey Yekhanin. Locally decodable codes: a brief survey. In *International Conference on Coding and Cryptology*, pages 273–282. Springer, 2011.

[168] John R. Zaller. *The Nature and Origins of Mass Opinion*. Cambridge University Press, 1992.