April 2021

Concentration Inequalities in the Wild: Case Studies in Blockchain & Reinforcement Learning

A. Pinar Ozisik
University of Massachusetts Amherst

CONCENTRATION INEQUALITIES IN THE WILD: CASE STUDIES IN BLOCKCHAIN & REINFORCEMENT LEARNING

A Dissertation Presented

by

A. PINAR OZISIK

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February, 2021

College of Information and Computer Sciences

CONCENTRATION INEQUALITIES IN THE WILD: CASE STUDIES IN BLOCKCHAIN & REINFORCEMENT LEARNING

A Dissertation Presented

by

A. PINAR OZISIK

Approved as to style and content by:

Brian Neil Levine, Chair

Philip S. Thomas, Member

Yuriy Brun, Member

Phillipa Gill, Member

Nikunj Kapadia, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

ACKNOWLEDGMENTS

It takes an army, not just a village, to complete the long, arduous, and exciting but frustrating journey that is the Ph.D. Many thanks first and foremost go to my advisor, Brian Levine, for being an incredibly supportive mentor and collaborator, and for having the monk-like patience to deal with my struggles and contentions. I learned so much from you. Thank you Philip Thomas for the countless hours you have spent teaching and guiding me in a completely new field. You are rocking this new role as a mentor. Thank you Phillipa Gill for the support and mentoring you provided towards the end of my Ph.D. I wish you had been around even sooner! I also thank the rest of my committee, Yuriy Brun and Nikunj Kapadia, for their thoughtful feedback on this work.

I am very grateful to my collaborators, George Bissias, Gavin Andresen, Amir Houmansadr and Sunny Katkuri, for their guidance in formulating a research agenda, and sharing their vast knowledge. Thank you to all the professors, especially Gordon Anderson and Bill Verts, for whom I have been a teaching assistant. By imparting your wisdom on how to be a great teacher, you have helped me shape my own teaching philosophy. I am also thankful to my undergraduate mentors, Antonella D. Lillo, for being a strong and determined role model, and Kyle Harrington, for introducing me to the curious world of research. I also want to thank Leeanne Leclerc and Emma Anderson for helping me achieve the milestones toward the Ph.D.

In addition to the people who have supported me professionally, I am also very grateful to those who were by my side emotionally. Thank you Marc Liberatore and Eva Hudlicka for listening to and encouraging me when I was at my lowest. To all my friends at UMass, thank you for including me in a great community of friends

and colleagues. I was inspired by each and every one of you. Thank you to my tribe of wonderful and inspirational women: Asli, Defne, Ece, Eren Sila, Zeynep, with the addition of Naz and Sila Ozkara. It has been a privilege to grow up together and transform into the women we are today; I appreciate you all. Joey—your unexpected debut into my life has been a breath of fresh air.

My number one supporters, my grandmothers, both of whom I lost during this journey—thank you for teaching me the importance of kindness and staying true to myself. And of course, my family: Mom, Dad and sister Deno, who have been with me from start to finish, and have unconditionally supported me through it all—my thank yous for you are uncountable.

ABSTRACT

CONCENTRATION INEQUALITIES IN THE WILD: CASE STUDIES IN BLOCKCHAIN & REINFORCEMENT LEARNING

FEBRUARY, 2021

A. PINAR OZISIK

B.S., BRANDEIS UNIVERSITY

B.A., BRANDEIS UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Brian Neil Levine

Concentration inequalities (CIs) are a powerful tool that provide probability bounds on how a random variable deviates from its expectation. In this dissertation, first I describe a blockchain protocol that I have developed, called `Graphene`, which uses CIs to provide probabilistic guarantees on performance. Second, I analyze the extent to which CIs are robust when the assumptions they require are violated, using Reinforcement Learning (RL) as the domain.

`Graphene` is a method for interactive set reconciliation among peers in blockchains and related distributed systems. Through the novel combination of a Bloom filter and an Invertible Bloom Lookup Table, `Graphene` uses a fraction of the network bandwidth used by deployed work for one- and two-way synchronization. It is a

fast and implementation-independent algorithm that uses CIs for parameterizing an IBLT so that it is optimal in size for a given desired decode rate. I characterize performance improvements through analysis, detailed simulation, and deployment results for Bitcoin Cash, a prominent cryptocurrency. Implementations of `Graphene`, IBLTs, and the IBLT optimization algorithm are all open-source code.

Second, I analyze the extent to which existing methods rely on accurate training data for a specific class of RL algorithms, known as Safe and Seldonian RL. Several Seldonian RL algorithms have a component called the safety test, which uses CIs to lower bound the performance of a new policy with training data collected from another policy. I introduce a new measure of security to quantify the susceptibility to corruptions in training data, and show that a couple of Seldonian RL methods are extremely sensitive to even a few data corruptions, completely breaking the probability bounds guaranteed by CIs. I then introduce a new algorithm, called `Panacea`, that is more robust against data corruptions, and demonstrate its usage in practice on some RL problems, including a grid-world and diabetes treatment simulation.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

LIST OF ALGORITHMS

CHAPTER 1

INTRODUCTION

Randomness is inherent in the world. We owe our unique existence to random biological processes, mutation and recombination, that result in the genetic composition of our DNA. However, in our everyday lives, the uncertainty of our future due to the random nature of our world, especially nowadays escalated by the onset of a worldwide pandemic, can be daunting, and even frightening. Yet in computer science, we manage to use randomness as an asset. This dissertation discusses two significant instances where randomness is valuable to computation.

1.1 Randomness as a Resource

At times, computer scientists study naturally occurring phenomena using probability and statistics, which model how randomness works; at other times, they artificially add randomness to algorithms. In computer networking, the Ethernet protocol leverages randomness by requiring each node in a network to wait a random amount of time before transferring frames in order to avoid collisions [83]. In cryptography, a random key and algorithm encode information, enabling a sender and receiver to privately communicate over the Internet. In operating systems, lottery scheduling addresses the problem of starvation, by incorporating randomness into an algorithm used to select processes. In the following section, we discuss the role of randomness as it pertains to specific problems in blockchain systems [87, 113] and *Seldonian reinforcement learning* [108].

1.1.1 Randomness for Set Reconciliation in Blockchains

The first half of this dissertation attempts to solve a canonical problem in networking and distributed systems, known as *set reconciliation*, to achieve synchronization among replicas, in the context of blockchain systems. Our version of the problem poses the question of how to relay information, i.e., a set of items, from a sender to a receiver if the receiver already possesses all or some of the items. First, we create a protocol assuming that the receiver has all of the items, and then extend it to the case where the receiver is missing items.

Our proposed solution, instead of sending all the items directly, leverages randomness by using *probabilistic data structures*. These data structures use hash functions to compactly represent a set of items. Hence, our introduced source of randomness comes from these hash functions, given the widely held assumption that a good hash function appears random [83]. The benefit of our non-deterministic protocol is minimized bandwidth given a low error rate, introduced by the use of probabilistic data structures.

Minimizing bandwidth while guaranteeing performance is crucial in blockchain systems for numerous reasons, which are discussed in detail in Section 2.1. To summarize briefly, the majority of bandwidth in these systems is due to *transactions*, similar to those generated by financial institutions. Reduced bandwidth implies that peers in the system can receive transactions more rapidly, and encourages the participation of peers with limited-bandwidth links. Furthermore, the system can scale up if it can support more transactions.

1.1.2 Randomness for Safety in Seldonian RL

The second half of this dissertation analyzes a class of *reinforcement learning* (RL) algorithms, referred to as *Safe and/or Seldonian RL* [108]. This class of algorithms uses randomly sampled datasets to make predictions, and ensures *safety*, which guarantees

that the algorithm will behave correctly with a user-specified probability. It is a specific component, called the *safety test*, that is largely responsible for ensuring safety. Our analysis of this component answers to what extent the safety guarantee holds when a subset of the samples is not random.

In real world applications, such non-random samples are caused by anomalies in training data. For example, RL has been suggested for type 1 diabetes treatment wherein training data includes measurements taken from a patient's insulin pump [108]. These devices, often with Internet connection, can run out of battery or malfunction. Marin et al. [76] showed how a malicious attacker could exploit the security protocols of a peacemaker to harm a person wearing the device. As more medical devices with Internet access get approved for distribution, such vulnerabilities will only increase. Therefore, we must address how to handle discrepancies, which can lead to very dangerous consequences, in data.

Our formulation of the problem represents anomalies as samples artificially created by a malicious attacker. The incorporation of an adversary provides a worst-case analysis to understand the robustness of safety tests to anomalies in data; because any algorithm robust to an adversary is also robust to non-adversarial anomalies in data. After quantifying the robustness of existing methods, we introduce a new algorithm, `Panacea`, which provides a user-specified level of robustness when the number of non-random samples in the dataset is upper bounded.

Next, we define a *concentration inequality* (CI), and provide a brief history. Then we present some canonical applications of CIs in computer science to exhibit their importance and benefits, before specifying how they are used in blockchain systems and Seldonian RL.

1.2 Concentration Inequalities

Both sections of the dissertation leverage CIs to bound the probability that a random variable deviates from some value, typically its expectation or median. A CI is a proven statement that specifies the properties of the random variable, and also quantifies the following: 1) The specific value around which the values of the random variable are centered; 2) The deviation or distance around this center; and 3) The probability of observing values within a given distance to the center. Such statements are also known as *tail inequalities* or *tail bounds*. A random variable is highly concentrated if its values are close to some specific value with high probability.

All CIs require samples to be random, but differ in terms of the additional information needed in order to use them. Some considerations for picking the right CI for an application domain include whether random variables are independent and drawn from a specific distribution, as well as whether it is easy to obtain a sample mean and variance.

1.2.1 A Brief History

Mathematicians, especially those interested in probability theory, statistical mechanics and functional analysis—subareas dealing with an infinite number of variables—studied and developed ideas surrounding the concentration of a random variable within the last century [15]. In 1952, the American statistician Herman Chernoff published a paper, which introduced the canonical form of the Chernoff bound. However, in his personal essay titled "A career in statistics," which appeared in the book, "Past, Present and Future of Statistics," Chernoff regretfully claimed that he should have also given credit to his colleague Herman Rubin, a statistician, who earned his Ph.D. from the University Chicago in 1948 [67]. Alas, it is Rubin's proof of Chernoff's upper bound that is often presented to graduate students today.

Rubin used Markov's inequality to obtain Chernoff's upper bound. In fact, Markov's inequality is also a misnomer because it first appeared in the work of Russian mathematician Pafnuty Chebyshev [50]. Although it is named after Chebyshev's student, Andrey Markov, sometimes it is also referred to as the first Chebyshev inequality (making the well-known Chebyshev inequality the second Chebyshev inequality). In his proof, Rubin applied the Markov inequality to the *moment generating function*—a set of mathematical functions uniquely characterizing a probability distribution—of a random variable.

A major development occurred with the birth of a concept called the *concentration of measure*, which appeared in a 1971 paper, by mathematician Vitali Milman [63]. French mathematician, Michel Talagrand, expanded on Milman's work in the 1990's and formally defined this concept, which he summarized as follows: "A random variable that depends (in a 'smooth' way) on many independent random variables (but not too much on any of them) is essentially constant" [15, 103]. Depending on the smoothness condition, this statement can have multiple meanings [15]. Nonetheless, it often means that "a random variable X concentrates around" the sample mean "$\overline{x}$ in a way that the probability of the event $\{|X - \overline{x}| \geq t\}$, for a given $t > 0$, decays exponentially in t" [97]. In the decades following Talagrand's work, CIs gained momentum, becoming a main subject of study and widely used in areas such as statistics, physics and the social sciences.

1.2.2 Application of Concentration Inequalities

In computer science, CIs are important tools with many applications. The subfield of randomized algorithms is dedicated to solving difficult problems, by adding a non-deterministic component into an algorithm, which outputs a random variable. Applying a CI to the observed samples, obtained over multiple runs of an algorithm, enables making predictions on the mean or median of that random variable. Additionally, CIs

provide a method to analyze the runtime of a randomized algorithm, i.e., how many runs are required such that the observed values predict the mean or median with high probability.

Another domain for the application of CIs in theoretical computer science is streaming algorithms. In the streaming model, at a high level, the goal is to compute some statistic—such as the mean, number of distinct numbers, most occurring number—from an input sequence of numbers within a given range, coming in one at a time [22]. A CI informs the design of a randomized algorithm that can compute the statistic with high probability, given space constraints, which are often defined as a function of the sequence size or the size of the universe of numbers from which the sequence is generated. Both in the case of streaming and randomized algorithms, CIs offer speed and simplicity, compared to their deterministic counterpart [86].

In machine learning, the assumption that samples are random is crucial to learn from data [1]. CIs offer a partial solution to the learning problem, which asks how to learn an unknown function f from a dataset [1]. Although a CI does not tell us what f is, it enables us to learn something, such as the mean or median of f, outside of the data [1].

1.2.2.1 Chernoff Bound for Blockchain Systems

In Chapter 2, we use the well-known *Chernoff* bound [23] for parameterizing our probabilistic data structures in order to guarantee our protocol's performance with high probability. The parameterization of our data structures ensures minimal bandwidth given a user-specified performance.

There are many variants of the Chernoff bound, depending on the the distribution of the random variable, and whether the random variable in question is a sum of other independent random variables. The version, which we utilize in Chapter 2, is

concerned with a random variable X that is the sum of n independent Bernoulli trials. The inequality states the following.

Chernoff Bound. Let X be the sum of n independent Bernoulli trials, $X_1, \ldots, X_n$, with mean $\mu = \mathbf{E}[X]$. Then for $\delta > 0$, $\Pr[X \geq (1+\delta)\mu] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu$.

For the same set of assumptions and variables, we use the following version of the bound:

$$\Pr[X \geq (1+\delta)\mu] \leq \mathrm{Exp}\left(-\frac{\delta^2}{2+\delta}\mu\right). \tag{1.1}$$

The derivation of Eq. (1.1) starting from the well-known version of the bound is:

$$\begin{aligned}
\Pr[X \geq (1+\delta)\mu] \leq & \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu \\
= & \mathrm{Exp}\left(\ln\left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^\mu\right) \\
= & \mathrm{Exp}(\mu(\ln(e^\delta) - \ln((1+\delta)^{1+\delta}))) \\
= & \mathrm{Exp}(\mu(\delta - (1+\delta)\ln(1+\delta))) \\
\leq & \mathrm{Exp}\left(\mu\left(\delta - (1+\delta)\left(\frac{2\delta}{2+\delta}\right)\right)\right) \\
= & \mathrm{Exp}\left(\mu\left(\frac{\delta(2+\delta) - 2\delta(1+\delta)}{2+\delta}\right)\right) \\
= & \mathrm{Exp}\left(\frac{-\delta^2}{2+\delta}\mu\right).
\end{aligned}$$

Above, we rely on the inequality that $\ln(1+x) \geq \frac{x}{1+x/2} = \frac{2x}{2+x}$ for $x > 0$ (see Love [71]), and that $e^{a-b} \leq e^{a-c}$ when $b \geq c$.

1.2.2.2 Chernoff-Hoeffding Bound for Seldonian RL

In Chapter 3, we analyze the affect of non-random samples added to a dataset, which is provided as input to a safety test that uses the *Chernoff-Hoeffding* (CH) [47] bound to guarantee safety. More accurately known as Hoeffding's inequality, this bound is named after the Finnish statistician, Wassily Hoeffding, who is regarded as one of the primary contributors to the subfield of nonparametric statistics [48].

His inequality is a generalization of the Chernoff bound, which initially considered a random variable that is the sum of Bernoulli random variables. In his proof written in 1963, Hoeffding relaxed the problem by bounding a random variable that is the sum of random variables that can come from any distribution. The inequality states the following.

Chernoff-Hoeffding Bound. Let $X_1, \ldots, X_n$ be independent bounded random variables with $X_i \in [a, b]$ for all n, where $-\infty < a \leq b < \infty$. Let $\overline{X} = {}^1\!/_n \sum_{i=1}^{n} X_i$ and $\mu = \mathbf{E}[X]$. Then for any $t > 0$, $\Pr(\overline{X} - \mu \geq t) \leq \mathrm{Exp}\left(- \frac{2nt^2}{(b-a)^2} \right)$.

The version of the CH inequality we use, keeping the same set of definitions and assumptions, is the following. For any $\delta \in [0, 1]$,

$$\Pr\left(X - (b-a)\sqrt{\frac{\ln(1/\delta)}{2n}} \geq \mu \right) \leq \delta. \tag{1.2}$$

To arrive at Eq. (1.2), we start from the canonical form of the CH inequality. Let $\delta = e^{-\frac{2nt^2}{(b-a)^2}}$. It follows that

$$\begin{aligned}
\frac{1}{\delta} =& e^{\frac{2nt^2}{(b-a)^2}} \\
\ln(1/\delta) =& \frac{2nt^2}{(b-a)^2} \\
t^2 =& \frac{\ln(1/\delta)(b-a)^2}{2n} \\
t =& \sqrt{\frac{\ln(1/\delta)(b-a)^2}{2n}} \\
=& (b-a)\sqrt{\frac{\ln(1/\delta)}{2n}}.
\end{aligned}$$

Replacing the definition of δ and t into the canonical form, we arrive at Eq. (1.2) .

$$\Pr(X - \mu \geq t) \leq \text{Exp}\bigg(- \frac{2nt^2}{(b-a)^2} \bigg)$$

$$\Pr(X - t \geq \mu) \leq \delta$$

$$\Pr\left(X - (b-a)\sqrt{\frac{\ln(1/\delta)}{2n}} \geq \mu \right) \leq \delta.$$

Overall, this dissertation provides a use case for the application of CIs, yet all the while determines their limitations when samples are not completely random—a likely scenario when anomalies such as errors, missing entries, malicious attacks etc. interfere with the natural course of events. Notice that randomness is necessary for the application of CIs, which, in the case of blockchain systems, reduces network bandwidth. In the case of Safe and Seldonian RL, it is due to the randomness of training data, that we are able to use CIs, and consequently, guarantee safety even when a subset of the samples is not random.

1.3 Collaborators

The study presented in Chapter 2 was conducted under the supervision of Professor Brian Levine. Additional collaborators include George Bissias, Gavin Andresen, Darren Tapp, Sunny Katkuri, and Amir Houmansadr [89, 90]. Chapter 3 was completed in collaboration with Philip Thomas [91], and was partly the result of insightful conversations with members of the Autonomous Learning Laboratory, specifically Yash Chandak and Stephen Giguere.

CHAPTER 2

SET RECONCILIATION APPLIED TO BLOCKCHAIN PROPAGATION

2.1 Introduction

Minimizing the network bandwidth required for synchronization among replicas of widely propagated information is a classic need of many distributed systems. Blockchains [87, 113] and protocols for distributed consensus [26, 56] are the most recent examples of systems where the performance of network-based synchronization is a critical factor in overall performance. Whether based on proof-of-work [55, 87], proof-of-stake [18, 39], or a directed acyclic graph (DAG) [65], the ability for these systems to scale to a large user base rely on assumptions about synchronization.

In all these systems, if the network protocol used for synchronization of newly authored *transactions* and newly mined *blocks* of validated transactions among peers is efficient, there are numerous benefits. First, if blocks can be relayed using less network data, then the maximum block size can be increased, which means an increase in the overall number of transactions per second. Scaling the transaction rate of Bitcoin is a critical performance issue [25, 27] driving many fundamental design choices such as inter-block time [113], block size, and layering [31]. Second, throughput is a bottleneck for propagating blocks larger than 20KB, and delays grow linearly with block size [25, 27]. As a consequence of the FLP result [35], blockchains cannot guarantee consensus. Smaller encodings of blocks allow for miners to reach consensus more rapidly, avoiding conflicts called *forks*. Moreover, systems based on GHOST [101], such as Ethereum [113], record forks on the chain forever, resulting in storage bloat.

Finally, using less bandwidth to relay a block allows greater participation by peers who are behind limited-bandwidth links or routes (e.g., China's firewall).

Contributions. In this chapter, we introduce `Graphene`, a probabilistic method and protocol for synchronizing blocks (and *mempools*) with high probability among peers in blockchains and related systems. `Graphene` uses a fraction of the network bandwidth of related work; for example, for larger blocks, our protocol uses 12% of the bandwidth of existing deployed systems. To do so, we make novel contributions to network-based *set reconciliation* methods and the application of probabilistic data structures to network protocols. We characterize our performance through analysis, detailed simulation, and open-source deployments. Our contributions include the following.

- We design a new protocol that solves the problem of determining which elements in a set M stored by a receiver are members of a subset $N \subseteq M$ chosen by a sender. We apply the solution to relaying a block of $n = |N|$ transactions to a receiver holding $m = |M|$ transactions. We use a novel combination of a Bloom filter [12] and an Invertible Bloom Lookup Table (IBLT) [44]. Our approach is smaller than using current deployed solutions [24] and previous IBLT-based approximate solutions [33]. We show our solution to this specific problem is an improvement of $\Omega(n \log n)$ over using an optimal Bloom filter alone.

- We extend our solution to the more general case where some of the elements of N are not stored by the receiver. Thus, our protocol extension handles the case where a receiver is missing transactions in the sender's block; our solution is a small fraction of the size of previous work [111] at the cost of an additional message. Additionally, we show how `Graphene` can efficiently identify transactions held by the receiver but not the sender.

- We design and evaluate an efficient search algorithm for parameterizing an IBLT so that it is optimally small in size but meets a desired decode rate with arbitrarily high probability. Because it is based on a hypergraph representation, it has faster execution times. This result is applicable beyond our context to any use of IBLTs.

- We design and evaluate a method for significantly improving the decode rate of an IBLT when two IBLTs are available that are based on roughly the same set of elements. This method is also a generally applicable.

- We provide a detailed evaluation using theoretical analysis and simulation to quantify performance against existing systems. We also characterize performance of our protocol in a live Bitcoin Cash deployment, and in an Ethereum implementation for historic blocks. We also show that `Graphene` is more resilient to attack than previous approaches.

We have publicly released our Bitcoin Cash and Ethereum implementations of `Graphene` [7, 53], a C++ and Python implementation of IBLTs including code for finding their optimal parameters [64], and we have released a public network specification of our basic protocol for standard interoperability [9]. It has been adopted by blockchain developers in released clients, replacing past approaches [10, 11]. While our focus is on blockchains, our work applies in general to systems that require set reconciliation, such as database or file system synchronization among replicas. Or for example systems such as CRLite [61], where a client regularly checks a server for revocations of observed certificates.

2.2 Background and Related Work

Below, we summarize and contrast related work in network-based set reconciliation and protocols for block propagation.

2.2.1 Set Reconciliation Data Structures

Set reconciliation protocols allow two peers, each holding a set, to obtain and transmit the union of the two sets. This synchronization goal is distinct from set membership protocols [21], which tell us, more simply, if an element is a member of a set. However, data structures that test set membership are useful for set reconciliation. This includes Bloom filters [12], a seminal probabilistic data structure with myriad applications [16, 72, 104]. Bloom filters encode membership for a set of size n by inserting the items into a small array of $\frac{-n \log_2(f)}{\ln(2)}$ bits. Each item is inserted $k = log(2)m/n$ times using independent hash functions. This efficiency gain comes at the expense of allowing a false positive rate (FPR), f.

Invertible Bloom Lookup Tables (IBLTs) [44] are a richer probabilistic data structure designed to recover the *symmetric difference* of two sets of items. Like Bloom filters, items are inserted into an IBLT's array of c *cells*, which is partitioned into subsets of size c/k. Each item is inserted once into each of the k partitions, at indices selected by k hash functions. Rather than storing only a bit, the cells store aggregates of the actual items. Each cell has a *count* of the number of items inserted and the xor of all items inserted (called a *keySum*). The following algorithm [33] recovers the symmetric difference of two sets. Each set is stored in an IBLT, A and B, respectively, (with equal c and k values). For each pairwise cell of A and B, the keySums are xor'ed and the counts subtracted, resulting in a third IBLT: $A \triangle B = C$ that lacks items in the intersection. The cells in C with count $= 1$ hold an item belonging to only A, and to only B if count $= -1$. These items are removed from the $k - 1$ other cells, which decrements their counts and allows for the additional *peeling* of new items. This process continues until all cells have a count of 0. (A *checkSum* field catches a special case: if x values in a cell from A and $x - 1$ values that are not a subset from the corresponding cell in B are subtracted, then count$= 1$ but the value contained is part of neither set.) If c is too small given the actual symmetric difference, then

iterative peeling will eventually fail, resulting in a *decode failure*, and only part of the symmetric difference will be recovered.

There are many variations of Bloom filters that present different trade-offs, such as more computation for smaller size. Similarly, IBLTs are one of several alternatives. For example, several approaches involve more computation but are smaller in size [30, 80, 114] (see [33] for a comparison). We have not investigated how alternatives to IBLTs improve `Graphene`'s size nor how, for example, computational costs differ. Our focus is on IBLTs because they are balanced: minimal computational costs and small size. While miners may have strong computational resources, *full nodes* and lighter clients in blockchains may not. More importantly, as our deployment results in Section 2.5.3 show, `Graphene`'s size grows slowly as block size increases for the most likely scenarios in Bitcoin, Bitcoin Cash, and Ethereum, demonstrating that IBLTs are a good fit for our problem. Finally, some of these solutions are complementary; for example, minsketch [114] can be used within the cells of IBLTs to reduce `Graphene`'s size further.

Comparison to related work. We provide a novel solution to the problem of set reconciliation, where one-way or mutual synchronization of information is required by two peers. Our results are significantly better than deployed past work that is based on Bloom filters alone [111] or IBLTs alone [33, 44], as we show in Section 2.5.3.

Byers et al. [19] introduce a new data structure called Approximate Reconciliation Trees (ARTs), based on Merkle trees, for set reconciliation. Their focus is different than ours, as their goal is not to discover the exact symmetric difference between two sets held by sender and receiver. Eppstein et al. [33] also present a set reconciliation solution, based primarily on IBLTs. They show their approach is more efficient for symmetric differences of 10k or fewer than ARTs as well as Characteristic Polynomial Interpolation [80], which is another solution to set reconciliation. Our method is more

efficient than these past approaches for discovering the exact symmetric difference in terms of storage or computation or both.

We provide several contributions to IBLTs. In general, if one desires to decode sets of size j from an IBLT, a set of values $\tau > 0$ and $k > 2$ must be found that result in $c = j\tau$ cells (divisible by k) such that the probability of decoding is at least p. We provide an implementation-independent algorithm for finding values τ and k that meet rate p and result in the smallest value of c.

Our work advances the usability of IBLTs. Goodrich and Mitzenmacher [44] provide values of τ that *asymptotically* ensure a failure rate that decreases polynomially with j. But these asymptotic results are not optimally small in size for finite j and do not help us set the value of k optimally. Using their unreleased implementation, Eppstein et al. [33] identify optimal τ and k that meet a desired decode rate for a selection of j values; however, the statistical certainty of this optimality is unclear. In comparison, using our open-source IBLT implementation [64], we are able to systematically produce statistically optimal values for τ and k (within a finite search range) for a wide range of j values. Because our method is based on hypergraphs, it is an order of magnitude faster than this previous method [33].

We also contribute a novel method for improving the decode rate of IBLTs that is similar in approach to Gollakota and Katabi [40]. Our method is complementary to related work by Pontarelli et al. [95], who have the same goal.

2.2.2 Block Propagation

Blockchains, distributed ledgers, and related technologies require a network protocol for distributing new transactions and new blocks. Almost all make use of a p2p network, often a clique among *miners* that validate blocks, and a random topology among non-mining *full* nodes that store the entire chain. New transactions have an ID equal to their cryptographic hash. When a new transaction is received, a peer sends the

ID as the contents of an inventory (`inv`) message to all d neighbors, who request a `getdata` message if the transaction is new to them. Transactions are stored in a *mempool* until included in a valid block. Blocks are relayed similarly: an `inv` is sent to each neighbor (often the header is sent instead to save time), and a `getdata` requests the block if needed. The root of a Merkle tree [78] of all transactions validates an ordered set against the mined block.

The block consists of a header and a set of transactions. These transactions can be relayed by the sender in *full*, but this wastes bandwidth because they are probably already stored by the receiver. In other words, blocks can be relayed with a compressed encoding, and a number of schemes have been proposed. As stated in Section 2.1, efficient propagation of blocks is critical to achieving consensus, reducing storage bloat, overcoming network firewall bottlenecks, and allowing scaling to a large number of transactions per second.

Transactions that offer low fees to miners are sometimes marked as DoS spam and not propagated by full nodes; yet, they are sometimes included in blocks, regardless. To avoid sending redundant `inv` messages, peers keep track, on a per-transaction and per-neighbor basis, whether an `inv` has been exchanged. This log can be used by protocols to send missing transactions to a receiver proactively as the block is relayed.

Comparison to related work. *Xtreme Thinblocks* [111] (XThin) is a robust and efficient protocol for relaying blocks, and is deployed in Bitcoin Unlimited (BU) clients. The receiver's `getdata` message includes a Bloom filter encoding the transaction IDs in her mempool. The sender responds with a list of the block's transaction IDs shortened to 8-bytes (since the risk of collision is still low), and uses the Bloom filter to also send any transactions that the receiver is missing. XThin's bandwidth increases with the size of the receiver's mempool, which is likely a multiple of the block size. In comparison, `Graphene` uses significantly lower bandwidth both when the receiver is

and is not missing transactions. However, `Graphene` may use an additional roundtrip time to repair missing transactions.

Compact Blocks [24] is a protocol that is deployed in the Bitcoin Core, Bitcoin ABC, and Bitcoin Unlimited clients. In this protocol, the receiver's `getdata` message is a simple request (no Bloom filter is sent). The sender replies with the block's transaction IDs shorted to 6-bytes (as well as the coinbase transaction). If the receiver has missing transactions, she requests repairs with a followup `inv` message. Hence, the network cost is $6n$ bytes, which is smaller than XThin's cost of a Bloom filter and a list of transaction IDs, which is approximately $\frac{m \log_2(f)}{8ln(2)} + 6n$; however, when the receiver is missing transactions, Compact Blocks has an extra roundtrip time, which may cost more if enough transactions are missing. `Graphene` is significantly lower in cost than Compact Blocks, as we show in Section 2.5.3.

Recently, *Xthinner* [110] was proposed as a variant of Xthin that employs compression techniques on the list of transactions in a block. Since the author states that Xthinner is not as compact as `Graphene`, we do not compare against it [109].

2.3 The Graphene Protocol

The primary goal of `Graphene` is to reduce the amount of network traffic resulting from synchronization of a sender and receiver; we do so in the context of block propagation. To motivate `Graphene`, consider a protocol that uses a Bloom filter alone to encode a block containing n transactions. Assume the receiver has a mempool of m transactions that are a super set of the sender's block. If we set the FPR of the sender's Bloom filter to $f = \frac{1}{144(m-n)}$, then we can expect the filter to falsely include an extra transaction in a relayed block about once every 144 blocks (approximately once a day in Bitcoin). This approach requires $\frac{-n \log_2(f)}{8 \ln(2)}$ bytes, and it is easy to show that it is smaller than Compact Blocks ($6n$ bytes) when $m < 71,982,340 + n$, which is highly likely.

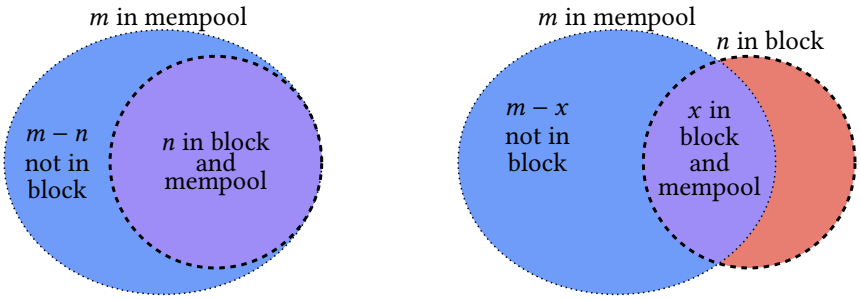

Figure 2.1: (Left) The receiver's mempool contains the entire block; *Protocol 1:* `Graphene` manages this scenario. (Right) The receiver's mempool does not contain the entire block. *Protocol 2:* `Graphene Extended` manages this scenario.

But we can do better than using a Bloom filter alone: in `Graphene`, we shrink the size of the sender's Bloom filter by increasing its FPR, and we correct any false positives at the receiver with an IBLT. The summed size of the two structures is smaller than using either alone. In practice, our technique performs significantly better than Compact Blocks for all but the smallest number of transactions, and we show in Section 2.5.3 that it performs better than *any* Bloom-filter-only approach asymptotically.

We design two protocols for `Graphene`, which we define presently. Both protocols use probabilistic data structures that fail with a tunable probability. Throughout our exposition, we use the concept of probabilistic *assurance*. Specifically, a property A is said to be held in data structure X with β-assurance whenever it is possible to tune X so that A occurs in X with probability at least β.

In **Protocol 1**, we assume that the receiver's mempool contains all transactions in the block, a typical case due to the aggressive synchronization that blockchains employ. So, our first design choice is to optimize for this scenario illustrated in Fig. 2.1-Left. As we show in Section 2.5.3, Protocol 1 is usually enough to propagate blocks.

In **Protocol 2**, we do not assume that the receiver's mempool is synchronized, as illustrated in Fig. 2.1-Right, which allows us to apply it to two scenarios: *(i)* block relaying between unsynchronized peers; and *(ii)* intermittent mempool synchronization.

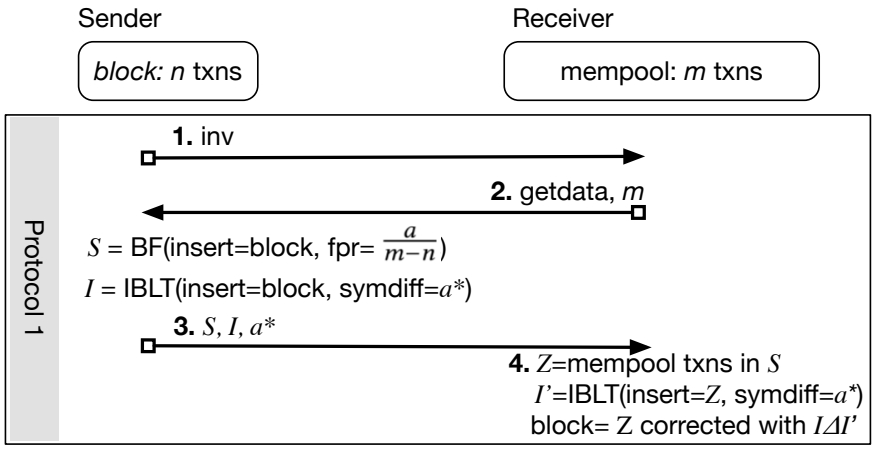

Figure 2.2: An illustration of Protocol 1 for propagating a block that is a subset of the mempool.

A receiver may not be synchronized with the sender because of network failures, slow transaction propagation times relative to blocks, or if the block contains unpropagated low-fee transactions erroneously filtered out as spam. Protocol 2 begins when Protocol 1 fails: the receiver requests missing transactions using a second Bloom filter; and the sender transmits any missing transactions, along with a second IBLT to correct mistakes. (Compact Blocks and XThin also handle this scenario but do so with greater network bandwidth.)

2.3.1 Protocols

Our first protocol is for receivers whose mempool contains all the transactions in the block; see Fig. 2.1-Left. The protocol is illustrated in Fig. 2.2.

PROTOCOL 1: Graphene

1: Sender: The sender transmits an `inv` (or blockheader) for a block.

2: Receiver: The receiver requests the unknown block, including a count of transactions in her mempool, m.

3: Sender: The sender creates Bloom filter **S** and IBLT **I** from the transaction IDs of the block (purple area in Fig. 2.1-Left). The FPR of **S** is $f_S = \frac{a}{m-n}$, and the

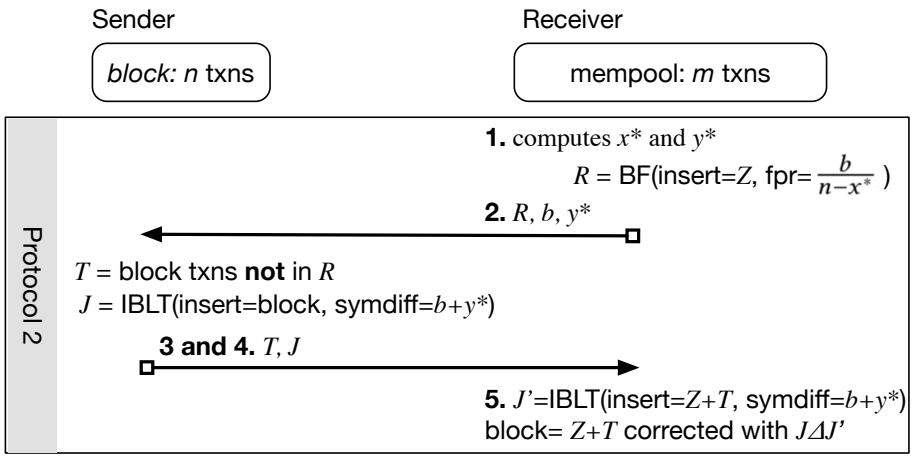

Figure 2.3: If Protocol 1 fails (e.g., if the block is not a subset of the mempool), Protocol 2 recovers with one roundtrip.

IBLT is parameterized such that a^* items can be recovered, where $a^* > a$ with β-assurance (outlined in green in Fig. 2.4). We set a so as to minimize the total size of **S** and **I**. **S** and **I** are sent to the receiver along with the block header (if not sent in Step 1).

4: Receiver: The receiver creates a candidate set Z of transaction IDs that pass through **S**, including false positives (purple and **dark blue** areas in Fig. 2.4). The receiver also creates IBLT $\mathbf{I}'$ from Z. She *subtracts* $\mathbf{I} \triangle \mathbf{I}'$, which evaluates to the symmetric difference of the two sets [33]. Based on the result, she adjusts the candidate set, validates the Merkle root in the block header, and decodes the block.

In blockchains, the sender knows the transactions for which no `inv` message has been exchanged with the receiver[1]; those transactions could be sent at Step 3 in order to reduce the number of transactions in $\mathbf{I} \triangle \mathbf{I}'$. (N.b., the IBLT stores only 8 bytes of each transaction ID; but full IDs are used for the Bloom filter.)

[1]In theory peers know this information; but in practice they use lossy data structures to keep track of this information.

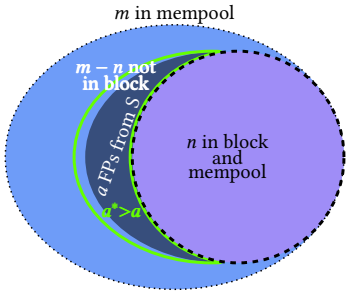

Figure 2.4: [Protocol 1] Passing m mempool transactions through $\mathbf{S}$ results in a FPs (in **dark blue**). A green outline illustrates $a^* > a$ with β-assurance, ensuring IBLT $\mathbf{I}$ decodes.

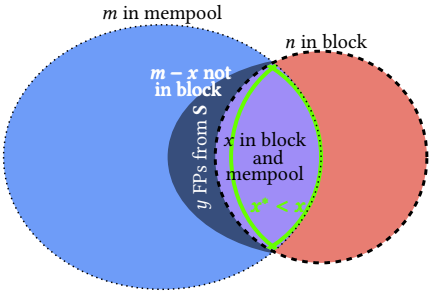

Figure 2.5: [Protocol 2] Passing m transactions through $\mathbf{S}$ results in z positives, obscuring a count of x TPs (**purple**) and y FPs (in **dark blue**). From z, we derive $x^* < x$ with β-assurance (in green).

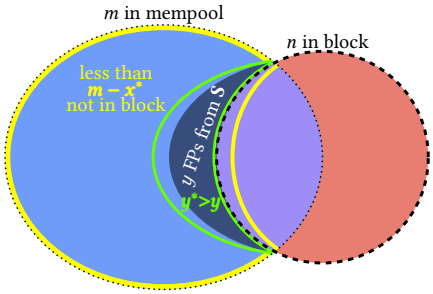

Figure 2.6: [Protocol 2] From our bound $m - x^* > m - x$ with β-assurance (in yellow), we can derive a bound for the false positives from $\mathbf{S}$ as $y^* > y$ with β-assurance outlined in green.

We use a fast algorithm to select a such that the total amount of data transmitted over the network is optimally small; see Section 2.3.3.1. The count of false positives from $\mathbf{S}$ has an expected mean of $(m-n)f_S = a$, whose variance comes from a Binomial distribution with parameters $(m-n)$ and f_S. Because of this variance, a^* should be used to parameterize $\mathbf{I}$ instead of a. We derive a^* in Section 2.3.3.1 via a Chernoff bound.

2.3.2 Graphene Extended

If the receiver does not have all the transactions in the block (Fig.2.1-Right), IBLT subtraction in Protocol 1 will not succeed. In that case, the receiver should continue with the following protocol, illustrated in Fig. 2.3. Subsequently, we show how this

protocol can also be used for intermittent mempool synchronization. Our contribution is not only the design of this efficient protocol, but the derivation of parameters that meet a desired decode rate.

PROTOCOL 2: Graphene Extended

1: Receiver: The size of the candidate set is $|Z| = z$, where $z = x+y$, a sum of x true positives and y false positives (**purple** and **dark blue** areas in Fig. 2.5). Because the values of x and y are obfuscated within the sum, the receiver calculates x^* such that $x^* \leq x$ with β-assurance (green outline in Fig. 2.5) She also calculates y^* such that $y^* \geq y$ with β-assurance (green outline in Fig. 2.6).

2: Receiver: The receiver creates Bloom filter $\mathbf{R}$ and adds all transaction IDs in Z to $\mathbf{R}$. The FPR of the filter is $f_R = \frac{b}{n-x^*}$, where b minimizes the size of $\mathbf{R}$ and IBLT $\mathbf{J}$ in step 4. She sends $\mathbf{R}$, y^* and b.

3: Sender: The sender passes all transaction IDs in the block through $\mathbf{R}$. She sends all transactions that are not in $\mathbf{R}$ directly to the receiver (red area of Fig. 2.6)

4: Sender: The sender creates and sends an IBLT $\mathbf{J}$ of all transactions in the block such that $b + y^*$ items can be recovered from it. This size accounts for b, the number of transactions that falsely appear to be in $\mathbf{R}$, and y^*, the number of transactions that falsely appear to be in $\mathbf{S}$.

5: Receiver: The receiver creates IBLT $\mathbf{J}'$ from the transaction IDs in Z and the new transaction IDs sent by the sender in step 3. She decodes the *subtraction* of the two blocks, $\mathbf{J} \triangle \mathbf{J}'$. From the result, she adjusts set Z, validates the Merkle root, and decodes the block.

As in Protocol 1, we set b so that the summed size of $\mathbf{R}$ and $\mathbf{J}$ is optimally small; see Section 2.3.3.1. We also derive solutions for x^* and y^*; see Section 2.3.3.2.

2.3.2.1 Mempool Synchronization

With a few changes, Protocols 1 and 2 can be used by two peers to synchronize their mempools so that both parties obtain the union of the two mempools, instead of a block that is a subset of one of the mempools. In this context, instead of a block, the sender places his entire mempool in **S** and **I**. The receiver passes her mempool through **S**, adding any negatives to H, the set of transactions that are not in **S**. Some transactions that the sender does not have in his mempool will falsely pass through **S**, and these are identified by **I** (assuming that it decodes); these transactions are also added to H. If **I** does not decode, Protocol 2 is executed to find transactions in the symmetric difference of the mempools; all missing transactions among the sender and receiver are exchanged, including those in set H. The protocol is more efficient if the peer with the smaller mempool acts as the sender since **S** will be smaller. Section 2.5.3.2 shows that the protocol is efficient.

2.3.3 Ensuring Probabilistic Data Structure Success

Cryptocurrencies allow no room for error: the header's Merkle root can be validated with an exact set of transactions only. Yet, `Graphene` is a probabilistic solution, and if its failure rate is high, resources are wasted on recovery. In this section, we derive the parameters for `Graphene` that ensure a tunable, high success rate.

2.3.3.1 Parameterizing Bloom filter S and IBLT I

`Graphene` sends the least amount of data over the network when the sum of the Bloom filter **S** and IBLT **I** is minimal. Let $T = T_{BF} + T_I$ be the summed size of the Bloom filter and IBLT. The size of a Bloom filter in bytes, T_{BF}, with false positive

rate f_S and n items inserted is $T_{BF} = \frac{-n \ln(f_S)}{8 \ln^2 2}$ [12]. Recall that we recover up to a^* items from the IBLT, where $a^* > a$ with β-assurance. As we show in Section 2.3.3.1, $a^* = (1 + \delta)a$, where δ is parameterized by β. An IBLT's size is a product of the number of items recoverable from a symmetric difference and a multiplier τ that ensures recovery at a desired success rate. Therefore, given the cost of r bytes per cell, T_I is

$$T_I = r\tau(1 + \delta)a.$$

When we set $f_S = \frac{a}{m-n}$, the total size of the Bloom filter and IBLT in bytes is

$$T(a) = \frac{-n \ln(\frac{a}{m-n})}{8 \ln^2 2} + r\tau(1 + \delta)a. \tag{2.1}$$

The value of a that minimizes T is either: $a = 1$; $a = m - n$; or the value of a where the derivative of Eq. (2.1) with respect to a is equal to zero. When $\delta = 0$ this is equal to

$$a = n/(8r\tau \ln^2 2). \tag{2.2}$$

Eq. (2.2) is a good approximation for the minimum when δ is close to 0; and the exact value is difficult to derive. Furthermore, implementations of Bloom filters and IBLTs involve non-continuous ceiling functions. As a result, Eq. (2.2) is accurate only for $a \geq 100$; otherwise the critical point a' produced by Eq. (2.2) can be inaccurate enough that $T(a')$ is as much as 20% higher than its true minimum value. `Graphene` exceeds the performance of previous work when Eq. (2.2) is used to select a. However, implementations that desire strictly optimal performance should take an extra step. If Eq. (2.2) results in a value of a less than 100, its size should be computed using

accurate ceiling functions and compared against all points $a < 100$. This is a typical case in our implementation for current block sizes.

Derivation of a^*. We can parameterize IBLT $\mathbf{I}$ based on the expected number of false positives from $\mathbf{S}$, but to ensure a high decode rate, we must account for the natural variance of false positives generated by $\mathbf{S}$. Here we derive a closed-form expression for a^* as a function of a and β such that $a^* > a$ holds with β-assurance, i.e. $a^* > a$ with probability at least β. Define $A_1, \ldots, A_{m-n}$ to be independent Bernoulli trials such that $Pr[A_i = 1] = f_S$, $A = \sum_{i=1}^{m-n} A_i$, and $\mu = E[A]$.

Theorem 1. *Let m be the size of a mempool that contains all n transactions from a block. If a is the number of false positives that result from passing the mempool through Bloom filter $\boldsymbol{S}$ with FPR f_S, then $a^* \geq a$ with probability β when*

$$a^* = (1+\delta)a,$$
$$\textit{where } \delta = \frac{1}{2}(s + \sqrt{s^2 + 8s}) \textit{ and } s = \frac{-\ln(1-\beta)}{a}.$$

Proof. There are $m - n$ potential false positives that pass through $\mathbf{S}$. They are a set $A_1, \ldots, A_{m-n}$ of independent Bernoulli trials such that $Pr[A_i = 1] = f_S$. Let $A = \sum_{i=1}^{m-n} A_i$ and $\mu = E[A] = f_S(m-n) = \frac{a}{m-n}(m-n) = a$. From Lemma 1, we have

$$Pr[A \geq (1+\delta)\mu] \leq \text{Exp}\left(-\frac{\delta^2}{2+\delta}\mu \right),$$

for $\delta \geq 0$. The receiver can set a bound of choice, $0 < \beta < 1$, and solve for δ using the right hand side of the above equation. To bound with high probability, we seek the complement of the right hand side

$$\begin{aligned} \beta &= 1 - \mathrm{Exp}\Big(- \tfrac{\delta^2}{2+\delta} a \Big) \\ \delta &= \tfrac{1}{2}(s + \sqrt{s^2 + 8s}), \text{ where } s = \tfrac{-\ln(1-\beta)}{a}. \end{aligned} \tag{2.3}$$

□

According to Theorem 1, if the sender sends a Bloom filter with FPR $f_S = \frac{a}{m-n}$, then with β-assurance, no more than a^* false positives will be generated by passing elements from Z though **S**. To compensate for the variance in false positives, IBLT **I** is parameterized by a symmetric difference of $a^* = (1+\delta)a$ items. **I** will decode subject to its own error rate (see Section 2.4), provided that $a < a^*$ (which occurs with probability β) and the receiver has all n transactions in the block. We evaluate this approach in Section 2.5.3; see Fig. 2.16.

2.3.3.2 Parameterizing Bloom filter R and IBLT J

Parameterizing *b*. In Protocol 2, we select b so that the summed size of **R** and **J** is optimally small. Its derivation is similar to a. We show below that $y^* = (1+\delta)y$. Thus, for protocol 2 the total size is:

$$T(b) = \frac{z \ln(\frac{b}{n-x^*})}{8 \ln^2 2} + r\tau(1+\delta)b. \tag{2.4}$$

When $\delta = 0$, the optimal value of b assuming continuous values is

$$b = z/(8r\tau \ln^2 2). \tag{2.5}$$

Similar to Section 2.3.3.1, an exact closed form for b is difficult to derive; and a perfectly optimal implementation would compute $T(b)$ using ceiling functions for values of $b < 100$.

Using z to parameterize R and J. Here we offer a closed-form solution to the problem of parameterizing Bloom filter **R** and IBLT **J**. This is a more challenging problem because x and y cannot be observed directly.

Let z be the observed count of transactions that pass through Bloom filter **S**. We know that $z = x + y$: the sum of x true positives and y false positives, illustrated as purple and **dark blue** areas respectively in Fig. 2.5. Even though x is unobservable, we can calculate a lower bound x^*, depending on x, z, m, f_S and β, such that $x^* \leq x$ with β-assurance, illustrated as a green outline in Fig. 2.5.

With x^* in hand, we also have, with β-assurance, an upper bound on the number of transactions the receiver is missing: $n - x^* > n - x$. This bound allows us to conservatively set $f_R = \frac{b}{n-x^*}$ for Bloom filter **R**. In other words, since $x^* < x$ with β-assurance, the sender, using **R**, will fail to send no more than b of the $n - x$ transactions actually missing at the receiver. IBLT **J** repairs these b failures, subject to its own error rate (see Section 2.4).

We also use x^* to calculate, with β-assurance, an upper bound $y^* \geq y$ on the number of false positives that pass through **S**. The green area in Fig. 2.6 shows y^*, which is a superset of the actual value for y, the **dark blue** area.

The sender's IBLT **J** contains all transactions in the block. The receiver's IBLT $\mathbf{J}'$ contains true positives from **S**, false positives from **S**, and newly sent transactions. Therefore, we bound both components of the symmetric difference by $b+y^*$ transactions in order for the subtraction operation to decode. In other words, both **J** and $\mathbf{J}'$ are parameterized to account for more items than actually exist in the symmetric difference between the two IBLTs. Note that we use β-assurance to bound the performance of

each probabilistic data structure; in doing so, we establish worst-case performance guarantees for our protocol.

The following theorems derive values for x^* and y^*.

Theorem 2. *Let m be the size of a mempool containing $0 \leq x \leq n$ transactions from a block. Let $z = x + y$ be the count of mempool transactions that pass through $\boldsymbol{S}$ with FPR f_S, with true positive count x and false positive count y. Then $x^* \leq x$ with probability β when*

$$x^* = \arg\min_{x^*} \; Pr[x \leq x^*; z, m, f_S] \leq 1 - \beta.$$

$$\textit{where} \;\; Pr[x \leq k; z, m, f_S] \leq \sum_{i=0}^{k} \left(\frac{e^{\delta_k}}{(1+\delta_k)^{1+\delta_k}} \right)^{(m-k)f_S}$$

$$\textit{and } \delta_k = \frac{z-k}{(m-k)f_S} - 1.$$

Proof. Let $Y_1, \ldots, Y_{m-x}$ be independent Bernoulli trials representing transactions not in the block that might be false positives; i.e., $Pr[Y_i = 1] = f_S$. We have $y = E[Y]$ and $Y = \sum_{i=1}^{m-x} Y_i$.

For a given value x, we can compute $Pr[Y \geq y]$, the probability of at least y false positives passing through the sender's Bloom filter. We apply a Chernoff bound [23]:

$$Pr[y; z, x, m] = \qquad Pr[Y \geq (1+\delta)\mu] \leq \left(\frac{e^{\delta}}{(1+\delta)^{1+\delta}} \right)^{\mu} \tag{2.6}$$

where $\delta > 0$, and $\mu = E[Y] = (m-x)f_S$. By setting $(1+\delta)\mu = z - x$ and solving for δ, we have

$$(1+\delta)(m-x)f_S = z - x$$

$$\delta = \frac{z-x}{(m-x)f_S} - 1.$$

We substitute δ into Eq. (2.6) and bound the probability of observing a value of $y = z - x$ or greater, given that the receiver has x transactions in the block. This

realization allows us to enumerate all possible scenarios for observation z. The cumulative probability of observing y, parametrized by z, given that the receiver has at most k of the transactions in the block, is:

$$\begin{aligned} Pr[x \leq k; z, m, f_S] &= \sum_{i=0}^{k} Pr[y; z, k, m] \\ &\leq \sum_{i=0}^{k} \left(\frac{e^{\delta_k}}{(1+\delta_k)^{1+\delta_k}} \right)^{(m-k)f_S} \end{aligned}$$

where $\delta_k = \frac{z-k}{(m-k)f_S} - 1$. Finally, using this closed-form equation, we select a bounding probability β, such as $\beta = 239/240$. We seek a probability β of observing z from a value x^* or larger; equivalently, we solve for the complement:

$$\arg\min_{x^*} Pr[x \leq x^*; z, m, f_S] \leq 1 - \beta.$$

To summarize, x^* is the *smallest* number of true positives such that the cumulative probability of observing $y = z - x^*$ false positives is at least $1 - \beta$. □

For good measure, we validated the theorem empirically, as shown in Fig. 2.7.

Theorem 3. *Let m be the size of a mempool containing $0 \leq x \leq n$ transactions from a block. Let $z = x + y$ be the count of mempool transactions that pass through $\boldsymbol{S}$ with FPR f_S, with true positive count x and false positive count y. Then $y^* \geq y$ with probability β when*

$$y^* = (1+\delta)(m - x^*)f_S,$$
$$\textit{where } \delta = \frac{1}{2}(s + \sqrt{s^2 + 8s}) \textit{ and } s = \frac{-\ln(1-\beta)}{(m-x^*)f_S}.$$

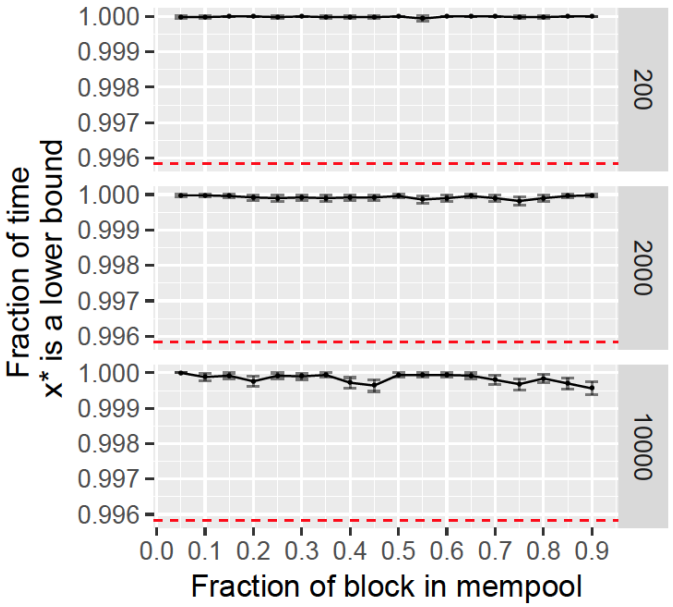

Figure 2.7: [Simulation, Protocol 2] The fraction of Monte Carlo experiments where $x^* < x$ via Theorem 2 compared to a desired bound of $\beta = 239/240$ (shown as a red dotted line).

Proof. First, we solve for $x^* \leq x$ with β-assurance using Theorem 2. We find $y^* = z - x^* \geq y$ by applying Lemma 1 to $Y = \sum_{i=1}^{m-x^*}$, the sum of $m - x^*$ independent Bernoulli trials such that $Pr[Y_i = 1] = f_S$ trials and $\mu = (m - x^*)f_S$:

$$Pr[Y \geq (1+\delta)\mu] \leq \text{Exp}\left(-\frac{\delta^2}{2+\delta}\mu\right),$$

for $\delta \geq 0$. We select $0 < \beta < 1$, and solve for δ using the right hand side of Eq. (2.7). To bound with high probability, we seek the complement of the right hand side.

$$\beta = 1 - \text{Exp}\left(-\frac{\delta^2}{2+\delta}(m - x^*)f_S\right) \tag{2.7}$$

$$\delta = \frac{1}{2}(s + \sqrt{s^2 + 8s}), \text{ where } s = \frac{-\ln(1-\beta)}{(m-x^*)f_S}. \tag{2.8}$$

Then, we set

$$y^* = (1+\delta)(m - x^*)f_S.$$

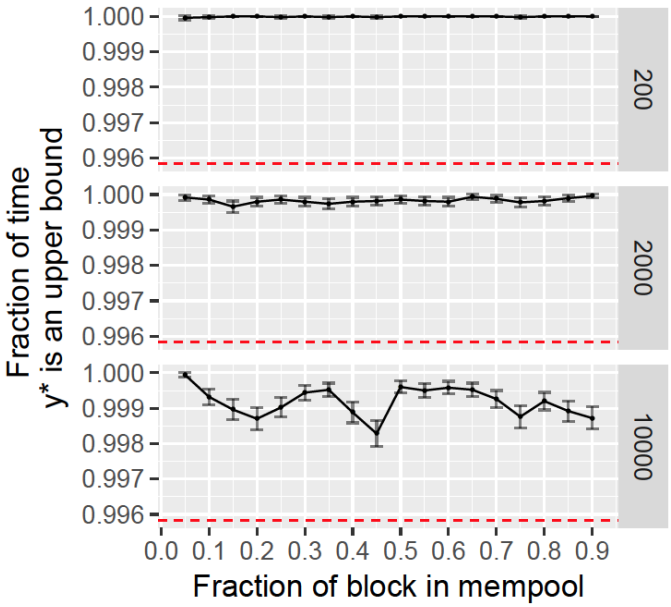

Figure 2.8: [Simulation, Protocol 2] The fraction of Monte Carlo experiments where $y^* > y$ via Theorem 3 compared to a desired bound of $\beta = 239/240$ (shown as a red dotted line).

Since, $x^* \leq x$ with β-assurance, it follows that y^* also bounds the sum of $m - x$ Bernoulli trials, where

$$y^* = (1 + \delta)(m - x)f_S,$$

with probability at least β for any $\delta \geq 0$ and $m > 0$. □

We validated this theorem empirically as well, as shown in Fig. 2.8.

Special case: $m \approx n$. When $m \approx n$, our optimization procedure in Protocol 1 will parameterize f_S to a value near 1, which is very efficient if the receiver has all of the block. But if $m \approx n$ and the receiver is missing some portion of the block, Protocol 1 will fail. Subsequently, with $z \approx m$, Protocol 2 will set $y^* \approx m$ and $x^* \approx 0$, and $f_R \approx 1$; and most importantly, IBLT $\mathbf{J}$ will be sized to m, making it larger than a regular block.

Fortunately, resolution is straightforward. If Protocol 1 fails, and the receiver finds that $z \approx m$, $y^* \approx m$, and $f_R \approx 1$, then in Step 2 of Protocol 2, the receiver should set f_R to a value less than 1. We set $f_R = 0.1$, but a large range of values execute efficiently (we tested from 0.001 to 0.2). All mempool transactions that pass through **S** are inserted into Bloom filter **R**, and **R** is transmitted to the sender.

The sender follows the protocol as usual, sending IBLT **J** along with h transactions from the mempool not in **R**. However, he then deviates from the protocol by also sending a third Bloom filter **F** intended to compensate for false positives from **R**. The $n-h$ transactions that pass through **R** are inserted into **F**. The roles of Protocol 2 are thus reversed: the sender uses Theorems 2 and 3 to solve for x^* and y^*, respectively, to bound false positives from **R** (substituting the block size for mempool size and f_R as the FPR). He then solves for b such that the total size in bytes is minimized for **F** with FPR $f_F = \frac{b}{m-x*}$ and **J** having size $b + y^*$. This case may be common when `Graphene` is used for mempool synchronization; our evaluations in Fig. 2.19 in Section 2.5.3.2 show that this method is more efficient than Compact Blocks.

Alternatives to Bloom filters. There are dozens of variations of Bloom filters [72, 104], including Cuckoo Filters [34] and Golomb Code sets [41]. Any alternative can be used if Eqs. (2.1), (2.2), (2.4), and (2.5) are updated appropriately.

2.4 Enhancing IBLT Performance

The success and performance of `Graphene` rests heavily on IBLT performance. Using IBLTs over a network has been studied in only a handful of papers [13, 33, 44, 81, 95], and current results are generally asymptotic with the size of the IBLT (the notable exception is Eppstein et al. [33], which we discuss in Section 2.2). In this section, we contribute several important results that allow for IBLTs to be used in practical systems with reliable precision. IBLTs are deceptively challenging to parameterize so that j items can be recovered with a desired success probability of

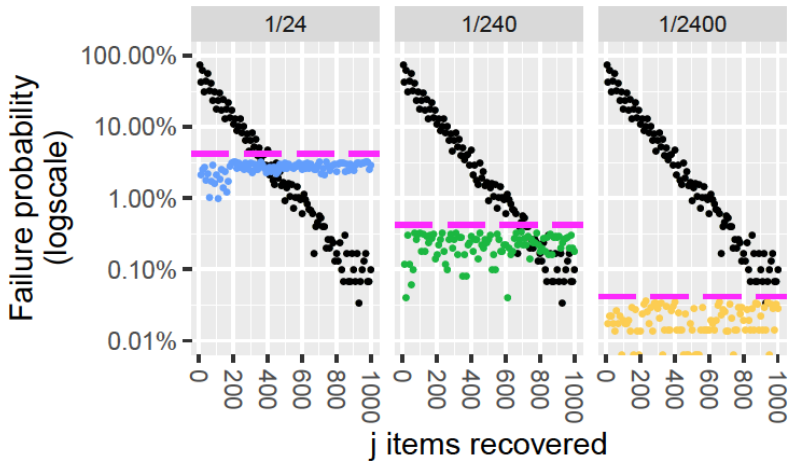

Figure 2.9: Parameterizing an IBLT statically results in poor decode rates. The black points show the decode failure rate for IBLTs when $k = 4$ and $\tau = 1.5$. The blue, green and yellow points show decode failure rates of optimal IBLTs, which always meet a desired failure rate on each facet (in magenta). Size shown in Fig. 2.11.

p, using the minimal number of cells. Only two parameters can be set: the *hedge* factor, τ (resulting in $c = j\tau$ cells total), and the number of hash functions, k, used to determine where to insert an item (each function covers c/k cells).

Motivation. Fig. 2.9 motivates our contributions, showing the poor decode rate of an IBLT if static values for k and τ are applied to small values of j. The figure shows three desired decode failure rates $(1 - p)$ in magenta: 1/24, 1/240, and 1/2400. The black points show the decode failure probability we observed in our IBLT implementation for static settings of $\tau = 1.5$ and $k = 4$. The resulting decode rate is either too small from an under-allocated IBLT, or exceeds the rate through over-allocation. The colored points show the failure rates of actual IBLTs parameterized by the algorithm we define below: they are optimally small and always meet or exceed the desired decode rate.

2.4.1 Optimal Size and Desired Decode Rate

Past work has never defined an algorithm for determining size-optimal IBLT parameters. We define an implementation-independent algorithm, adopting the IBLT

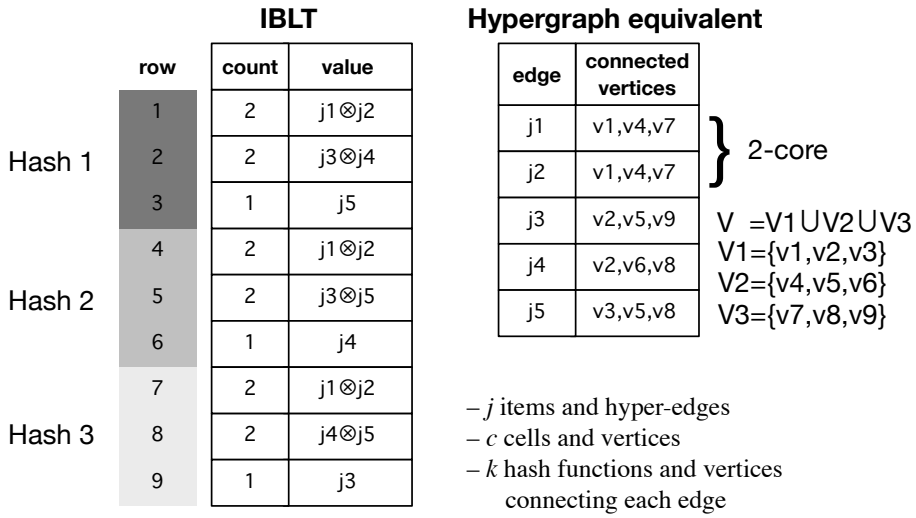

Figure 2.10: An example IBLT (without the checksum field) and its equivalent hypergraph representation. In the IBLT, $k = 3$, there are $c = 3k$ cells, and $j = 5$ items are placed in k cells. In the hypergraph, j hyperedges each have k vertices out of c vertices total.

interpretation of Molloy [84] and Goodrich and Mitzenmacher [44] as uniform hypergraphs.

Let $H = (V, X, k)$ be a *k-partite, k-uniform hypergraph*, composed of a set of c vertices. Let $V = V_1 \cup \cdots \cup V_k$, where each V_i is a subset of c/k vertices (we enforce that c is divisible by k). X is a set of j hyper-edges, each connecting k vertices, one from each of the V_i. The hypergraph represents an IBLT with k hash functions, j inserted items, and c cells. As illustrated in Figure 2.10, each cell corresponds to a vertex, and we have $|V| = c$ and $|V_i| = c/k$. Each item represents an edge connecting k vertices, with the ith vertex being chosen uniformly at random from V_i. Vertices in V_i correspond to hash function i, which operates over a distinct range of cells.

Decoding corresponds to removing edges that contain a vertex of degree 1, repeatedly. The *r-core* [99] of H is the maximal subgraph (after decoding) in which all vertices have degree at least r. H contains a non-empty 2-core iff the IBLT it represents cannot be decoded. In the example illustrated in Figure 2.10, edges j_3, j_4 and j_5 contain degree 1 vertices v_9, v_6, and v_3 respectively and can be removed; the

remaining subgraph comprised of j_1 and j_2 and degree 2 vertices v_1, v_4, and v_7 is a 2-core. Equivalent operations on the IBLT would show that items $j1$ and j_2 cannot be decoded.

Algorithm 1: `IBLT-Param-Search`(j, k, p)

$c_l = 1$
$c_h = c_{max}$
$trials = 0$
$success = 0$
$L = (1 - p)/5$
while $c_l \neq c_h$ **do**
  $trials$ += 1
  $c = (c_l + c_h)/2$
  if decode(j, k, c) **then**
    $success$ += 1
  end if
  `conf`=conf_int($success$, $trials$)
  $r = success/trials$
  if $r -$ `conf` $\geq p$ **then**
    $c_h = c$
  end if
  if $(r +$ `conf` $\leq p)$ **then**
    $c_l = c$
  end if
  if $(r -$ `conf` $> p - L)$ and $(r +$ `conf` $< p + L)$ **then**
    $c_l = c$
  end if
end while
return c_h

Algorithm 1. This algorithm searches for the optimally small size of $c = j\tau$ cells that decodes j items with decode success probability p (within appropriate confidence intervals) from an IBLT with k hash functions. `decode()` operates over a hypergraph rather than a real IBLT.

We seek an algorithm for determining the most space-efficient choice for c and k that is sufficient to ensure a decode rate of p for a fixed number of inserted items j. Items are inserted pseudo-randomly by applying the hash functions. Therefore, it makes sense to model the edge set X as a random variable. Define $\mathcal{H}_{j,p} = \{(V, X, k) \mid E[\texttt{decode}((V, X, k))] \geq p, |X| = j\}$, or the set of hypergraphs (V, X, k) on

j edges whose expected decode success rate is bounded by p. Based on this definition, the algorithm should return

$$\underset{(V,X,k)\in\mathcal{H}_{j,p}}{\arg\min} \; |V|. \tag{2.9}$$

Our approach for solving Eq. (2.9) is to fix j, p, and k and perform binary search over all possible values for $c = |V|$. A key point is that binary search is justified by the fact that the expected decode failure rate is a monotonically increasing function of c. This notion can explained as follows. A 2-core forms in (V, X, k) when there exists some group of v vertices that exclusively share a set of at least $2v$ edges. Define vertex set U such that $|U| > |V|$. Since the j edges in X are chosen uniformly at random, and there are more possible edges on vertex set U, the probability that a given set of $2v$ edges forms in (U, X, k) must be lower than in (V, X, k).

Fig. 1 shows the pseudocode for our algorithm, which relies on two functions. The function `decode(j,k,c)` takes a random sample from the set of hypergraphs $\mathcal{H}_{j,p}$ and determines if it forms a 2-core (i.e., if it decodes), returning `True` or `False`. The function `conf_int(s,t)` returns the two-sided confidence interval of a proportion of s successes and t trials. We set $c_{max} = 20$ in our implementation; in general, that value could be made part of the search itself [6]. In practice, we call Alg. 1 from an outer loop of values of k that we have observed to be reasonable (e.g., 3 to 12), and prune the search of each k when it is clear that it will not be smaller in size than a known result. To be clear, the algorithm is optimal within the values of k that are searched. Selecting a maximum value of k to search is an open problem, but we observe a trend that smaller k are better as j increases. See Bissias [8] for a related discussion.

We have released an open-source implementation of IBLTs in C++ with a Python wrapper [64]. The release includes an implementation of Alg. 1 and optimal parameters for several decode rates. Although the runtime is linear with j, for any given rate, the parameter file can be generated once ever and be universally applicable to any IBLT implementation. Compared to a version of our algorithm that uses actual IBLTs, our

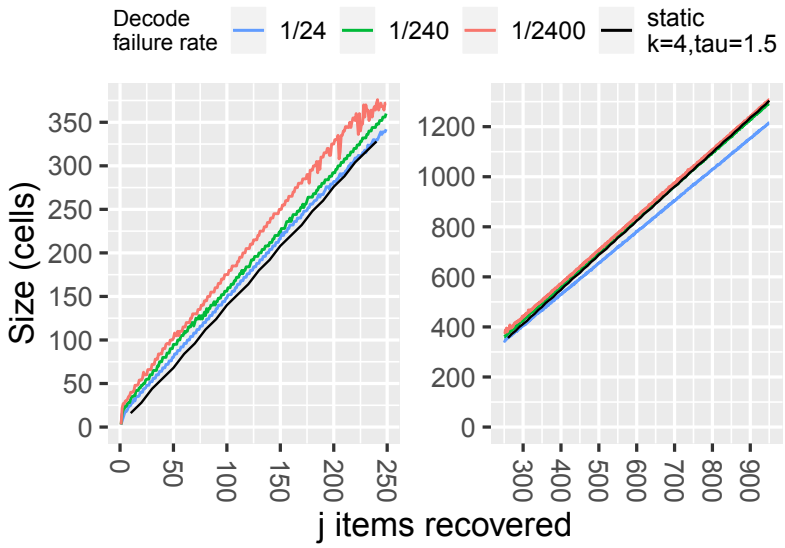

Figure 2.11: Size of optimal IBLTs (using Alg. 1) given a desired decode rate; with a statically parameterized IBLT ($k = 4, \tau = 1.5$) in black. For clarity, the plot is split on the x-axis. Decode rates are shown in Fig. 2.9.

hypergraph approach executes much faster for all j. For example, to parameterize $j = 100$, our approach completes in 29 seconds on average (100 trials). Allocating actual IBLTs increases average run time to 426 seconds. The speed increase is due to our use of hypergraphs, which are larger than IBLTs in terms of storage but are faster to compute with.

Fig. 2.11 shows the size of IBLTs when parameterized optimally for three different decode rates. If parameterized appropriately, the number of cells in an IBLT grows linearly, with variations due to inherent discretization and fewer degrees of freedom in small IBLTs.

2.4.2 Ping-Pong Decoding

`Graphene` takes advantage of its two IBLTs to increase the decode rate for Protocol 2. IBLTs $\mathbf{I}$ (and $\mathbf{I}'$) and $\mathbf{J}$ (and $\mathbf{J}'$) are different sizes, and may use a different number of hash functions, but contain the same transactions. When an IBLT fails to decode completely, it still can succeed partially. The transactions that are decoded from

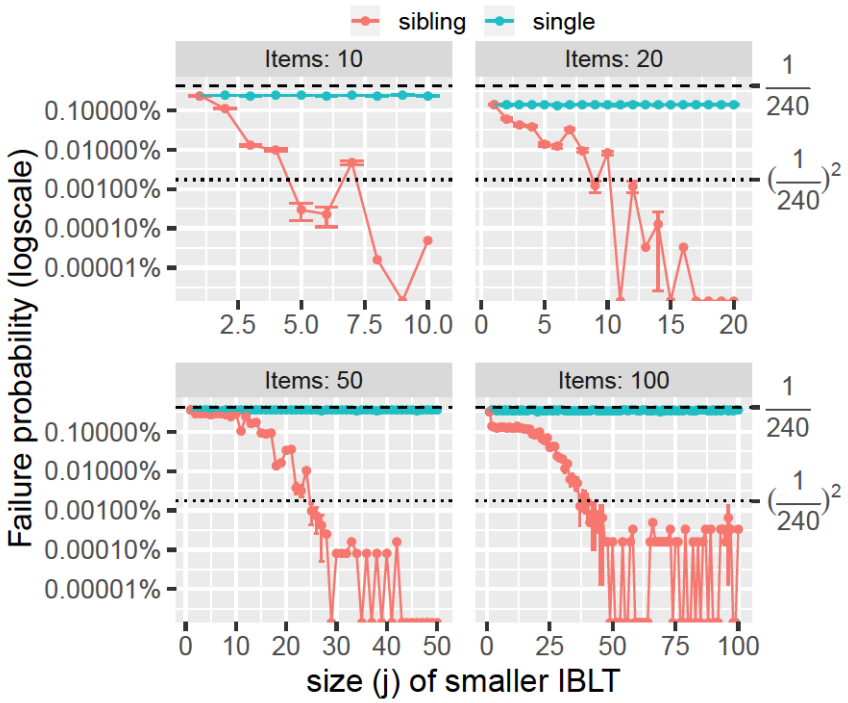

Figure 2.12: Decode rate of a single IBLT (parameterized for a 1/240 failure rate) versus the improved *ping-pong decode* rate from using a second, smaller IBLT with the same items.

$\mathbf{J} \triangle \mathbf{J}'$ can be removed from $\mathbf{I} \triangle \mathbf{I}'$, and decoding of the latter can be retried. Then, transactions from $\mathbf{I} \triangle \mathbf{I}'$ can be removed from $\mathbf{J} \triangle \mathbf{J}'$, and decoding of the latter can be retried; and so on in a *ping-pong* fashion (see Gollakota and Katabi [40]). We note that if the count of a decoded item is 1, then it should be subtracted from the other IBLT; if the count is -1, then it should be added to the other IBLT. The IBLTs should use different seeds in their hash functions for independence.

Fig. 2.12 shows an experiment where we compared the decode rate of a single IBLT parameterized to be optimally small and recover $j \in [10, 20, 50, 100]$ items with decode failure rate of $1 - p = 1/240$. We then inserted the same items into a second IBLT parameterized to hold $0 < i \leq j$ items. When i is the same size as j, the failure rate is $(1-p)^2$ or lower. But improvements can be seen for values $i < j$ as well. When j is small, very small values of i improve the decode rate. For larger values of j, larger

values of i are needed for decoding. Ping-pong decoding is computationally fast since IBLT decoding itself is fast.

The use of ping-pong decoding on `Graphene` Protocol 2 is an improvement of several orders of magnitude; results are presented in Fig. 2.17 in Section 2.5.3.2.

This approach can be extended to other scenarios that we do not investigate here. For example, a receiver could ask many neighbors for the same block and the IBLTs can be jointly decoded with this approach.

2.5 Evaluation

Our evaluation reaches the following conclusions:

- `Graphene` Protocol 1 is more efficient than using a Bloom filter alone, by $\Omega(n \log n)$ bits. For all but small n, it is more efficient than deterministic solutions.

- We deployed Protocol 1 worldwide in Bitcoin Cash and show it performs as expected; and our implementation of Protocol 1 for Ethereum evaluated against historic data also shows expected gains.

- Using extensive Monte Carlo simulations, we show that `Graphene` Protocols 1 and 2 are always significantly smaller than Compact Blocks and XThin for a variety of scenarios, including mempool synchronization.

- In simulation, the decode success rates of `Graphene` Protocols 1 and 2 are above targeted values.

2.5.1 Comparison to Bloom Filter Alone

The information-theoretic bound on the number of bits required to describe any unordered subset of n elements, chosen from a set of m elements is $\lceil \log_2 \binom{m}{n} \rceil \approx n \log_2(m/n)$ bits [17]. Carter et al. also showed that an approximate solution to the

problem has a more efficient lower bound of $-n \log_2(f)$ bits by allowing for a false positive rate of f [21].

Because our goal is to address a restricted version of this problem, `Graphene` Protocol 1 is more efficient than Carter's bound for even an optimal Bloom filter alone. This is because `Graphene` Protocol 1 assumes all n elements (transactions) are stored at the receiver, and makes use of that information whereas a Bloom filter would not.

Theorem 4. *Relaying a block with n transactions to a receiver with a mempool (a superset of the block) of m transactions is more efficient with Graphene Protocol 1 than using an optimally small Bloom filter alone, when the IBLT uses $k \geq 3$ hash functions. The efficiency gains of Graphene Protocol 1 are $\Omega(n \log_2 n)$.*

Proof. We assume that $m = cn$ for some constant $c > 1$. Our proof is asymptotic. Thus, according to the law of large numbers, every value $\delta > 0$ (where δ is defined as in Theorem 1) is sufficient to achieve β-assurance when choosing values for a^*, x^*, and y^*. Accordingly, we may proceed under the assumption that $\delta = 0$; i.e., there is no need to lower the false positive rate of either Bloom filter to account for deviations because the observed false positive rate will always match its expected value asymptotically.

Let f, where $0 < f < 1$, be the FPR of a Bloom filter created in order to correctly identify $n \geq 1$ elements from a set of $m \geq 1$ elements. The size of the Bloom filter that has FPR f, with n items inserted, is $-n \log_2(f)$ bits [21]. Let $f = \frac{p}{m-n}$, where $0 < p < 1$. The expected number of false positives that can pass through the Bloom filter is $(m-n)\frac{p}{(m-n)} = p$. Since $0 < p < 1$, one out of every $1/p$ Bloom filters is expected to fail.

To correctly identify the same set of items, `Graphene` instead uses a Bloom filter with $f = \frac{a}{m-n}$, where we set $a = n/(r\tau)$ since the Bloom filter is optimal, and uses an IBLT with $a\tau$ cells (r bytes each) that decodes with probability p. The expected number of false positives that pass through `Graphene`'s Bloom filter is $(m-n)\frac{a}{(m-n)} = a$.

An IBLT with 1 to a items inserted in it decodes with probability $1 - p$. In other words, one out of every $1/p$ `Graphene` blocks is expected to fail.

The difference in size is

$$\begin{aligned}
&-n \log_2 \left(\frac{p}{m-n}\right) - \left(- n \log_2 \left(\frac{a}{m-n}\right) + ar\tau \right) \\
&= n \log_2(a/p) - ar\tau \\
&= n(\log_2 n + \log_2 {}^{1}\!/_{p\tau}) - 1) \\
&= n(\log_2 n + \Omega(\tau^{2-k})) \\
&= \Omega(n(\log_2 n)),
\end{aligned}$$

where Eq. 2.5.1 follows from Theorem 1 from Goodrich and Mitzenmacher [44], given that we have an IBLT with $k \geq 3$ hash functions. □

`Graphene` cannot replace all uses of Bloom filters, only those where the elements are stored at the receiver, e.g., set reconciliation.

As $m - n$ approaches zero, Protocol 1 shrinks its Bloom filter and approaches an IBLT-only solution. The special case where `Graphene` has an FPR of 1 is equivalent to not sending a Bloom filter at all; in that case, `Graphene` is as small as any IBLT-only solution, as expected. As the size of $m - n$ increases, `Graphene` is much smaller than sticking with an IBLT-only solution, which would have $\tau(m - n)$ cells.

`Graphene` is not always smaller than deterministic solutions. As we show in our evaluations below, for small values of n (about 50–100 or fewer depending on constants), deterministic solutions perform better. For larger values, `Graphene`'s savings are significant and increase with n.

We leave analytic claims regarding Protocol 2 for future work; however, below we empirically demonstrate its advantage over related work.

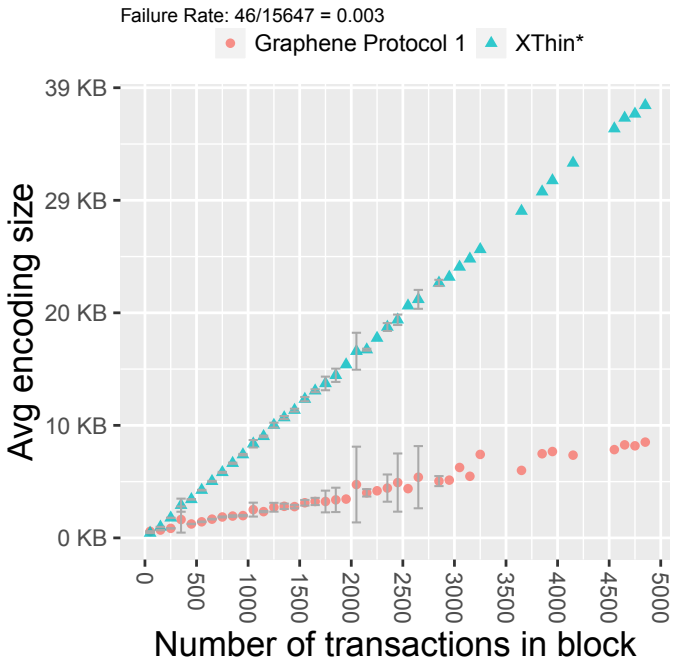

Figure 2.13: [Deployment on BCH, Protocol 1]: Performance of Protocol 1 as deployed on the Bitcoin Cash network, where the node was connected to 6 other peers. Points are averages of binned sizes; error bars show 95% c.i. if at least 3 blocks of that size can be averaged.

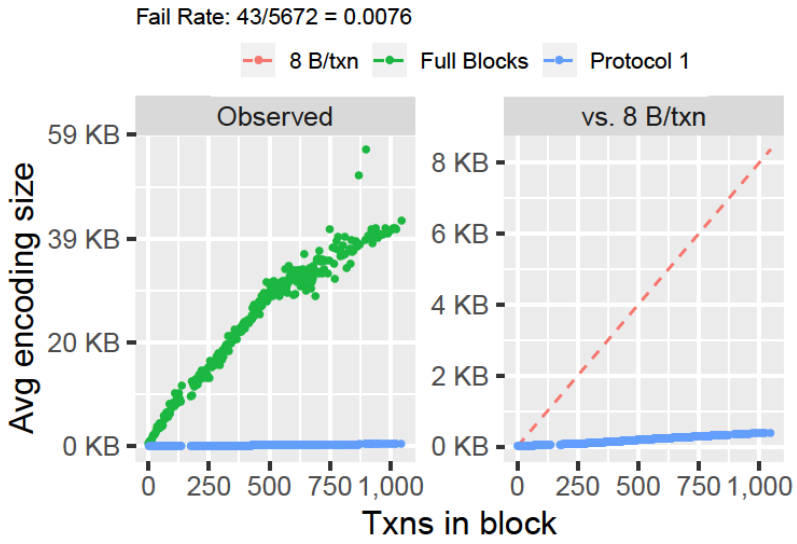

Figure 2.14: [Implementation, Protocol 1] An implementation of Protocol 1 for the Geth Ethereum client run on historic data. The left facet compares against Ethereum's use of full blocks; the right compares against an idealized version of Compact Blocks using 8 bytes/transaction.

2.5.2 Implementations

Bitcoin Cash implementation. We coded `Graphene` (Protocol 1) for Bitcoin Unlimited's Bitcoin Cash client. It appeared first in edition 1.4.0.0 (Aug 17, 2018) as an experimental feature, and since 1.5.0.1 (Nov 5, 2018) it has been the default block propagation technique. Currently, 686 nodes (operated by persons unknown to us) are running `Graphene` on the Bitcoin Cash mainnet.

Fig. 2.13 shows results from our own peer running the protocol on the real network from January 9–April 29, 2019. Fig. 2.13 also shows results from using Bitcoin Unlimited's XThin implementation; however, we have removed the cost of the receiver's Bloom filter to make the comparison fair (hence it is labelled *XThin**). The cost of XThin* is computed for the same blocks. As expected, while XThin* costs grow quickly, the costs of `Graphene` grow much more slowly as block size increases.

We have not yet deployed Protocol 2 (below we discuss our simulation of Protocol 2). Out of 15,647 `Graphene` blocks, 46 failed to decode, which is within our β-assurance of 239/240. This statistic also confirms our two-protocol approach: most of the time Protocol 1 is sufficient; and correcting failures with Protocol 2 is rarely needed if β is set appropriately. And although the failure rate of Protocol 1 is low, Protocol 2 is a necessary part of a complete solution for our probabilistic approach.

Ethereum implementation. We implemented `Graphene` Protocol 1 for *Geth*, Ethereum's primary client software, and submitted a Pull Request [53]. We replayed all the blocks produced on the Ethereum mainnet blockchain on Jan 14, 2019 (a total of 5,672 blocks), introducing new message types to comply with `Graphene`'s protocol. During our test, the size of the mempool at the receiver was kept constant at 60,000 transactions, which is typical (see `https://etherscan.io/chart/pendingtx`). The left facet of Fig. 2.14 shows the size in bytes of full blocks used by Ethereum and `Graphene`. The right facet compares `Graphene` (including transaction ordering infor-

mation) against a line showing 8 bytes/per transaction (an idealization of Compact Blocks without overhead).

2.5.3 Monte Carlo Simulation

Methodology and assumptions. We also wrote a custom block propagation simulator for `Graphene` (Protocols 1 and 2) that measures the network bytes exchanged by peers relaying blocks. We executed the protocol using real data structures so that we could capture the probabilistic nature of Bloom filters and IBLTs. Specifically, we used our publicly released IBLT implementation and a well-known Python Bloom filter implementation. In results below, we varied several key parameters, including the size of the block, the size of the receiver's mempool, and the fraction of the block possessed at the receiver. Each point in our plots is one parameter combination and shows the mean of 10,000 trials or more; if no confidence interval is shown, it was very small and removed for clarity. For all trials, we used a bound of $\beta = 239/240$ (see Eqs. (2.3) and (2.8)).

In all experiments, we evaluated three block sizes (in terms of transactions): 200, which is about the average size of Ethereum (ETH) and Bitcoin Cash (BCH) blocks; 2,000 which is the average size of Bitcoin (BTC) blocks; and 10,000 as an example of a larger block scenario. In expectation of being applied to large blocks and mempools, we used 8-byte transaction IDs for both `Graphene` and Compact Blocks. Also for Compact Blocks, we used `getdata` messages with block encodings of 1 or 3 bytes, depending on block size [24].

2.5.3.1 Graphene: Protocol 1

Size of blocks. Fig. 2.15 shows the cost in bytes of `Graphene` blocks compared to Compact Blocks. We focus on varying mempool size rather than block size. In

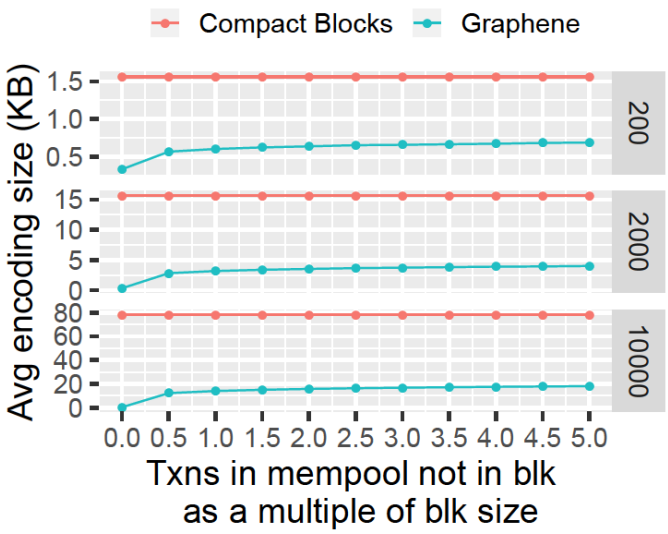

Figure 2.15: [Simulation, Protocol 1] Average size of `Graphene` blocks versus Compact Blocks as the size of the mempool increases as a multiple of block size. Each facet is a block size: (200, 2000, and 10000 transactions). (N.b., This figure varies mempool size; Fig. 2.13 varied block size.)

these experiments, the receiver's mempool contains all transactions in the block plus some additional transactions, which increase along the x-axis as a multiple of the block size. For example, at fraction 0.5 and block size 2,000, the mempool contains 3,000 transactions in total. The experiments demonstrate that `Graphene`'s advantage over Compact Blocks is substantial and improves with block size. Also, the cost of `Graphene` grows sublinearly as the number of extra transactions in the mempool grows.

Decode rate. Fig. 2.16 shows the decode rate of `Graphene` blocks, as the mempool size increases. In all cases, the decode rate far exceeds the desired rate, demonstrating that our derived bounds are effective. `Graphene`'s decode rate suffers when the receiver lacks the entire block in her mempool. For example, in our experiments, a receiver holding 99% of the block can still decode 97% of the time. But if the receiver holds less than 98% of the block, the decode rate for Protocol 1 is zero. Hence, Protocol 2 is required in such scenarios.

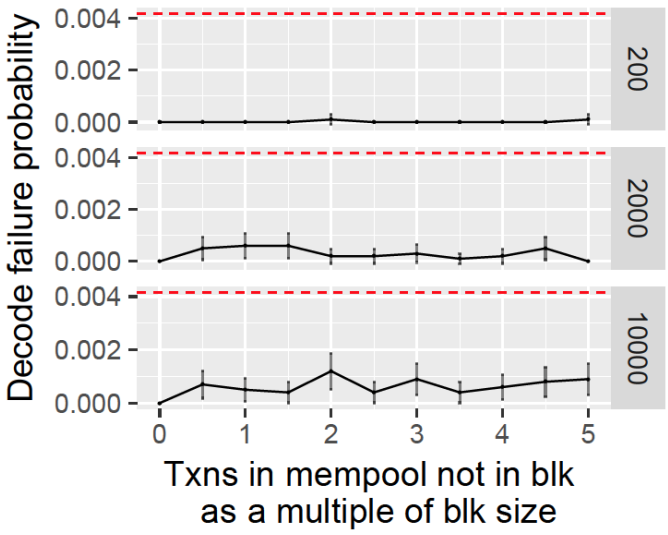

Figure 2.16: [Simulation, Protocol 1] Decode rate of `Graphene` blocks with $\beta = \frac{239}{240}$ (red dotted line), as block size and the number of extra transactions in the mempool increases as a multiple of block size.

2.5.3.2 Graphene Extended: Protocol 2

Our evaluations of Protocol 2 focus on scenarios where the receiver does not possess the entire block and $m > n$; we evaluate $m = n$ as a special case.

Size by message type. Fig. 2.18 shows the cost of `Graphene` Extended, broken down into message type, as the fraction of the block owned by the receiver increases. The dashed line on the plot shows the costs for Compact Blocks, where the receiver requests missing transactions by identifying each as a 1- or 3-byte index (depending on block size) in the original ordered list of transactions in the block encodings [24]. (We exclude the cost of sending the missing transactions themselves for both protocols.)

Overall, `Graphene` Extended is significantly smaller than Compact Blocks, and the gains increase as the block size increases. For blocks smaller than 200, eventually Compact Blocks would be smaller in some scenarios.

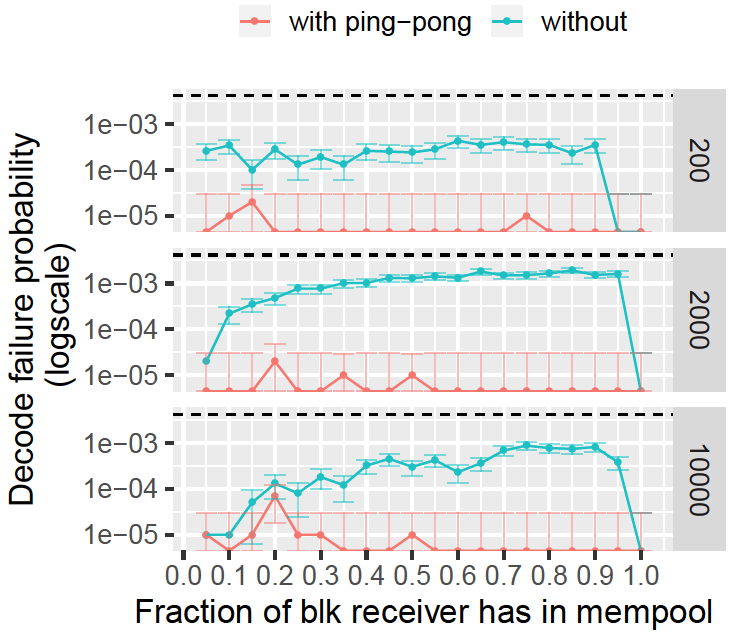

Figure 2.17: [Simulation, Protocol 2] Decode rate of `Graphene` blocks with $\beta = \frac{239}{240}$, shown by the black dotted line, as block size and the number of extra transactions in the mempool increase. Error bars represent 95% confidence intervals.

Decode rate and Ping-Pong enhancement. Fig. 2.17 shows the decode rate of `Graphene` blocks; it far exceeds the desired rate. And when ping-pong decoding is used, the simulation results show decoding rates close to 100%.

Not shown are our simulations of the Difference Digest by Eppstein et al. [33]. The Difference Digest is an IBLT-only solution that is an alternative to our Protocol 2. In that work, the sender begins by telling the receiver the value n. The receiver creates a Flajolet-Martin estimate [36] of $m - n$, using $\lceil \log_2(m-n) \rceil$ IBLTs, each with 80 cells where roughly m elements are inserted. The sender replies with a single IBLT of twice the number of cells as the estimate (to account for an under-estimate). This approach is several times more expensive than `Graphene`.

***m* ≈ *n* and mempool synchronization.** As described in Section 2.3.2.1, `Graphene` can be used for mempool synchronization, setting n to the size of the sender's mempool. In these cases, if peers are mostly synchronized, then $m \approx n$, which is a special case for

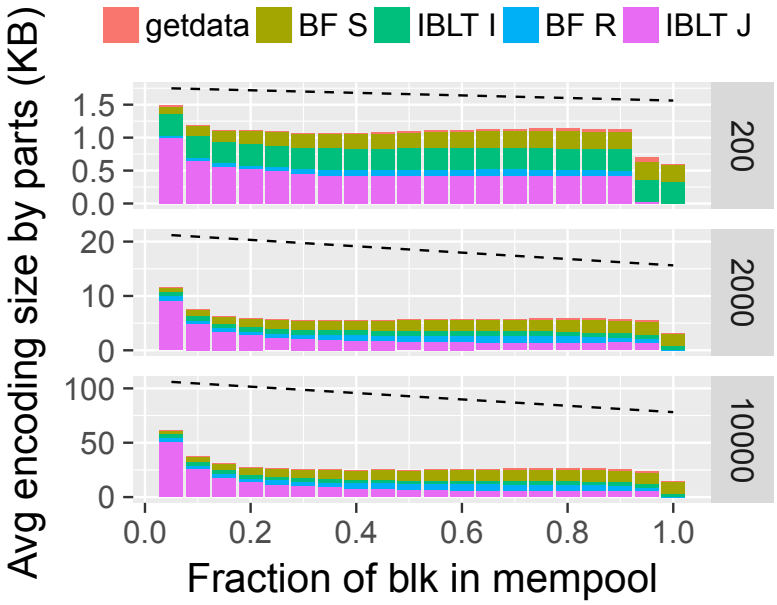

Figure 2.18: [Simulation, Protocol 2] `Graphene` Extended cost as the fraction of the block owned by the receiver increases. The black dotted line is the cost of Compact Blocks.

`Graphene` discussed in Section 2.3.3.1. Our evaluations of this scenario are shown in Fig. 2.19. In these experiments, the sender's mempool has n transactions, of which a fraction (on the x-axis) are in common with the receiver. The receiver's mempool size is topped off with unrelated transactions so that $m = n$. As a result, Protocol 1 fails and modifications from Section 2.3.3.1 are employed. As with previous experiments, `Graphene` performs significantly better than Compact Blocks across multiple mempool intersection sizes and improvement increases with block size.

2.6 Systems Issues

2.6.1 Security Considerations

Malformed IBLTs. It is simple to produce an IBLT that results in an endless decode loop for a naive implementation; the attack is just as easily thwarted. To create a malformed IBLT, the attacker incorrectly inserts an item into only $k - 1$ cells. When

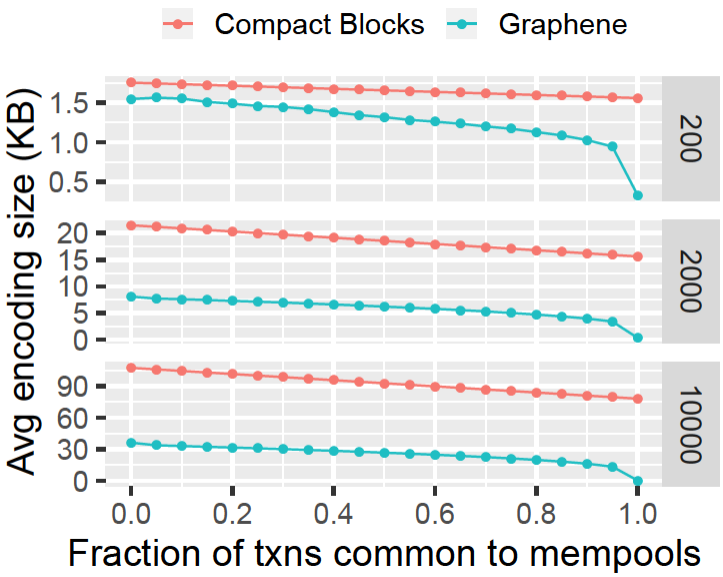

Figure 2.19: [Simulation, Mempool Synchronization] Here $m = n$ and the peers have a fraction of the sender's mempool in common on the x-axis. `Graphene` is more efficient, and the advantage increases with block and mempool size.

the item is peeled off, one cell in the IBLT will contain the item with a count of -1. When that entry is peeled, $k - 1$ cells will contain the item with a count of 1; and the loop continues. The attack is thwarted if the implementation halts decoding when an item is decoded twice. Once detected, the sender can be dropped or banned by the receiver.

Manufactured transaction collisions. The probability of accidental collision of two 8-byte transaction IDs in a mempool of size m is $\approx 1 - \text{Exp}\left(\frac{-m(m-1)}{2^{65}}\right)$ [82]. An attacker may use brute force search to discover and submit collisions. SipHash [3] is used by some blockchain protocols to limit the attack to a single peer.

With or without the use of SipHash, `Graphene` is more resilient against such collisions than XThin and Compact Blocks. Let t_1 and t_2 be transactions with IDs that collide with at least 8 bytes. In the worst case, the block contains t_1, the sender has never seen t_2, and the receiver possesses t_2 but has never seen t_1. In this case, XThin and Compact Blocks will always fail; however, `Graphene` fails with low

probability, $f_S \cdot f_R$. For the attack to succeed, first, t_2 must pass through Bloom filter **S** as a full 32-byte ID, which occurs only with probability f_S. If it does pass, the IBLT will decode but the Merkle root will fail. At this point, the receiver will initiate Protocol 2, sending Bloom filter **R**. Second, with probability f_R, t_1 will be a false positive in **R** as a full 32-byte ID and will not be sent to the receiver.

2.6.2 Transaction Ordering Costs

Bloom filters and IBLTs operate on unordered sets, but Merkle trees require a specific ordering. In our evaluations, we did not include the sender's cost of specifying a transaction ordering, which is $n \log_2 n$ bits. As n grows, this cost is larger than `Graphene` itself. Fortunately, the cost is easily eliminated by introducing a known ordering of transactions in blocks. In fact, Bitcoin Cash clients deployed a Canonical Transaction Ordering (CTOR) ordering in Fall 2018.

2.6.3 Reducing Processing Time

Profiling our implementation code revealed that processing costs are dominated heavily by passing the receiver's mempool against Bloom filter **S** in Protocol 1. Fortunately, this cost is easily reduced. A standard Bloom filter implementation will hash each transaction ID k times — but each ID is already the result of applying a cryptographic hash and there is no need to hash k more times; see Suisani et al. [102]. Instead, we break the 32-byte transaction ID into k pieces. Applying this solution reduced average receiver processing in our Ethereum implementation from 17.8ms to 9.5ms. Alternative techniques [28, 29, 54] are also effective and not limited to small values of k.

2.6.4 Limitations

`Graphene` is a solution for set reconciliation where there is a trade-off between transmission size, complexity (in terms of network round-trips), and success rate. In

contrast, popular alternatives such as Compact Blocks [24] have predictable transmission size, fixed transmission complexity, use a trivial algorithm, and always succeed. `Graphene`'s performance gains over related work increase as block size grows, but it is a probabilistic solution with a (tunable) failure rate. We do not claim to have the optimal solution for propagating blocks, nor for scaling blockchains in general.

CHAPTER 3

SECURITY ANALYSIS OF SAFE AND SELDONIAN REINFORCEMENT LEARNING ALGORITHMS

3.1 Introduction

Reinforcement learning (RL) algorithms have been proposed for many high-risk applications, such as improving type 1 diabetes and sepsis treatments [57, 108]. One type of *safe RL* algorithm [105, 107], subsequently referred to as *Safe and/or Seldonian RL* [108], enables these high-risk applications by providing high-confidence guarantees that the application will not cause undesirable behavior like increasing the frequency of dangerous patient outcomes.

However, existing safe RL algorithms rely on the assumption that training data is free from anomalies such as errors, missing entries, and malicious attacks. In real applications, anomalies are common when training data comes from a pipeline that includes human interactions, natural language processing, device malfunctions, etc. For example, the recent application of RL to sepsis treatment in the *intensive care unit* (ICU) used training data generated from hand-written doctors' notes [57]. In a high-stress ICU environment, missing records and poorly written notes are difficult to automatically parse [2]. Furthermore, Petit et al. [92] demonstrated the importance of using reliable training data for self-driving cars, a potential area for the real application of RL. They executed a series of attacks on the camera and sensors of self-driving cars in a lab environment to demonstrate how the safety of passengers can be compromised.

In this chapter, we analyze how robust Seldonian RL algorithms are to perturbations in data. Specifically, we analyze the robustness of a specific component, called the

safety test. This component makes current Seldonian algorithms safe: the safety test checks whether necessary safety constraints are satisfied with high probability. Using data collected from a baseline policy, it outputs new policies that are highly likely to perform at least as well as the baseline. The safety test first computes estimates of the expected performance of a new policy from training data, using *importance sampling* (IS). It then uses *concentration inequalities* (CI) to bound the expectation of the IS estimates.

First, we propose a new measure, which we call α-security, for quantifying how robust the safety test of a Seldonian RL algorithm is to data anomalies. To create this measure, we define an attacker that adds adversarially corrupt data points to training data. Although anomalies in data are often not due to an adversarial attacker, if we create algorithms that are robust to adversarial attacks, they will also be robust to non-adversarial anomalies in data. Second, we analyze the security of existing safety test mechanisms using α-security, and find that even if only one data point is corrupted, the high-confidence safety guarantees provided by several Seldonian RL algorithms can be egregiously violated. Then we propose a new algorithm that is more robust to anomalies in training data, ensuring safety with high probability when an upper bound on the number of adversarially corrupt data points is known. Finally, we present experiments that support our theoretical analysis.

Our work is directly applicable to any scenario that requires computing confidence intervals around IS estimates. More broadly, the community is also interested in our definition of safety [38] and its limitations [45], and IS [14, 51, 70, 75]. Lastly, our α-security formalization also pertains to high-confidence methods that do not use IS [60, 62, 105], and can be used as a general framework to study their robustness to data corruption attacks.

3.2 Background

A *Markov decision process* (MDP) is a mathematical model of the environment with which an agent interacts. Formally, it is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, d_0, \gamma)$. $\mathcal{S}$ is the set of possible states of the environment. S_t is the state of the environment at time $t \in \{0, 1, \dots\}$. $\mathcal{A}$ is the set of actions that an agent interacting with the environment can take. A_t is the action chosen by the agent at time t. For notational simplicity, we assume that $\mathcal{A}$ and $\mathcal{S}$ are finite.[1] $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the *transition function*, which characterizes the distribution of S_{t+1} given S_t and A_t, using the definition $\mathcal{P}(s, a, s') \coloneqq \Pr(S_{t+1}{=}s' | S_t{=}s, A_t{=}a)$. The reward provided to the agent at time t is a bounded real-valued random variable, R_t. The *reward function* $R : \mathcal{S} \times \mathcal{A} \to [R_{\min}, R_{\max}]$ captures sufficient information about the distribution of rewards given S_t and A_t in order to reason about optimal behavior, and is defined by $R(s, a) \coloneqq \mathbf{E}[R_t | S_t{=}s, A_t{=}a]$. The initial distribution of states is captured by $d_0 : \mathcal{S} \to [0, 1]$, i.e., $d_0(s) \coloneqq \Pr(S_0{=}s)$. Finally, $\gamma \in [0, 1]$ is a parameter used to discount rewards based on the time at which they occur.

We consider *episodic* MDPs, which contain a special state s_∞, called the *terminal absorbing state*. Once the agent enters state s_∞, it can never leave and all subsequent rewards are zero. Upon reaching s_∞, the trial, called an *episode*, has effectively ended because there are no more rewards to be obtained. Although in theory the agent will continue transitioning from s_∞ back to s_∞ forever, in practice, we can begin the next episode. We say that a problem has a *finite horizon*, τ, if $S_\tau = s_\infty$ almost surely, regardless of how actions are chosen. A *trajectory* H is the sequence of states, actions, and rewards from one episode: $H = (S_0, A_0, R_0, \dots, S_{\tau-1}, A_{\tau-1}, R_{\tau-1})$.

The mechanism for selecting actions within an agent is called a *policy*, which we denote by $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$, where $\pi(s, a) \coloneqq \Pr(A_t{=}a | S_t{=}s)$. We consider the batch

[1] Our work generalizes to continuous MDPs as well, but care must be taken to select π_e such that importance sampling weights are bounded.

RL setting wherein training data D, also referred to as the *safety data*, consists of trajectories generated using a single baseline policy π_b, called the *behavior policy*. The training data D, consists of n trajectories generated using π_b: $D \coloneqq \{H_i\}_{i=1}^n$. We write S_t^i, A_t^i, and R_t^i to denote the state, action, and reward at time t in the i^{th} trajectory in D. If $\mathcal{H}$ denotes the set of all possible trajectories, each policy induces a distribution over $\mathcal{H}$. We abuse notation by reusing π, and write $\text{supp}(\pi_b)$ to denote the support of this distribution. Let $\mathcal{H}_{\pi_b} = \text{supp}(\pi_b)$, i.e., the set of all trajectories that can be created by running π_b.

The *return* is the discounted sum of rewards, and the return from trajectory H is $g(H) \coloneqq \sum_{t=0}^{\tau-1} \gamma^t R_t$. The goal of the agent is to find a policy that maximizes the expected return it receives. This objective is captured by the objective function: $J(\pi) \coloneqq \mathbf{E}[\sum_{t=0}^{\tau-1} \gamma^t R_t | \pi]$, where conditioning on π denotes that $A_t \sim \pi(S_t, \cdot)$ for all t. To simplify notation later, we assume that $\sum_{t=0}^{\tau-1} \gamma^t R_t \in [0, 1]$.[2]

3.2.1 Safe Reinforcement Learning

Let a be a function that takes a dataset as input and produces a policy as output, i.e., $a(D)$ is the policy output by the algorithm a when run on training data D. Given a user-specified constant $\delta \in [0, 1]$ (typically $\delta = 0.05$ or $\delta = 0.01$), a *Seldonian* RL algorithm is any algorithm a that satisfies

$$\Pr(J(a(D)) \geq J(\pi_b)) \geq 1 - \delta. \tag{3.1}$$

That is, the algorithm guarantees that with probability at least $1 - \delta$, it will return a policy with expected return at least equal to that of the behavior policy. For simplicity, we assume that $J(\pi_b)$ is known—in previous work, $J(\pi_b)$, written instead as $\rho(\pi_b)$, was left as a user-specified constant, for example a high-confidence upper bound on $J(\pi_b)$.

[2]This is equivalent to assuming that rewards are bounded, since given a finite horizon, the returns can be normalized.

Note that an algorithm that always returns π_b is technically safe. However, our goal is to develop safe algorithms that frequently return policies that have larger expected return than π_b, while satisfying the safety guarantee in Eq. (3.1). In this framework, a user can define many different reward functions that improvement can be guaranteed with respect to (w.r.t.). This enables users to define safety constraints using reward functions. For discussion of how this provides a useful interface for defining safety constraints, see the work of Thomas et al. [108].

Several Seldonian RL algorithms have a component called the *safety test*, which ensures that Eq. (3.1) is satisfied [79, 107, 108]. The safety test takes three inputs: 1) A policy π_e that is evaluated for safety, referred to as the *evaluation policy*; 2) The safety data, D; and 3) $J(\pi_b)$. If there is sufficient confidence that $J(\pi_e) \geq J(\pi_b)$, the safety test returns `True`; otherwise, it returns `False`.

First, using each trajectory in D, the safety test computes n estimates of the expected performance of π_e. For this estimation, we make the standard assumption that $\pi_b(s,a) = 0$ implies $\pi_e(s,a) = 0$.[3] One method for estimating the expected value of a function when samples come from a different distribution (π_b) than the desired distribution (π_e) is importance sampling. In Seldonian RL, the safety test uses IS to produce an unbiased estimator of $J(\pi_e)$ from D [96]. Specifically, for each trajectory in D, it computes an *importance weighted return* that is defined by: $\hat{J}^{\star}(\pi_e|H_i,\pi_b) := g(H_i)w^{\star}(H_i,\pi_e,\pi_b)$, where $w^{\star}(H_i,\pi_e,\pi_b)$ is the *importance weight*. For traditional IS, $w^{\text{IS}}(H_i,\pi_e,\pi_b) := \prod_{t=0}^{\tau-1} \pi_e(A_t^i,S_t^i)/\pi_b(A_t^i,S_t^i)$. For *weighted importance sampling* (WIS) [73], $w^{\text{WIS}}(H_i,\pi_e,\pi_b) := n(\sum_{x=1}^{n}\prod_{t=0}^{\tau-1}\pi_e(A_t^x,S_t^x)/\pi_b(A_t^x,S_t^x))^{-1} \times \prod_{t=0}^{\tau-1}\pi_e(A_t^i,S_t^i)/\pi_b(A_t^i,S_t^i)$. Notice that WIS normalizes the importance weights using the sum of the importance weights in all trajectories. Given that $\pi_b(s,a) = 0$ implies $\pi_e(s,a) = 0$, IS is strongly consistent and unbiased, while WIS is strongly consistent,

[3] In the more general setting, where $\mathcal{A}$ and $\mathcal{S}$ are not finite, this assumption corresponds to requiring π_e to be absolutely continuous with respect to π_b.

but typically biased [106]. For the remainder of the chapter, we use $\star \in \{\text{IS, WIS}\}$ to denote weights computed using IS or WIS.

Second, the safety test uses concentration inequalities to bound the expectation of the importance weighted returns. A CI provides probability bounds on how a random variable deviates from its expectation. Let $\mathbf{X}$ denote the importance weighted returns obtained from the trajectories in D, i.e., $\mathbf{X} := \{X_i : i \in \{1, \ldots, n\}, X_i = \hat{J}^{\star}(\pi_e | H_i, \pi_b)\}$. Safety tests leverage CIs to lower bound the performance of π_e, using $\mathbf{X}$. A commonly used CI is the *Chernoff-Hoeffding* (CH) inequality [47], which states the following: For n independent, real-valued, and bounded random variables, $X_1, \ldots, X_n$, such that $\Pr(X_i \in [0, b]) = 1$ and $\mathbf{E}[X_i] = \mu$, for all $i \in \{1, \ldots, n\}$, where $b \in \mathbb{R}$, with probability at least $1 - \delta$, $\mu \geq {}^1\!/_n \sum_{i=1}^{n} X_i - b\sqrt{\ln({}^1\!/_\delta)/2n}$.

Let $L^{*,\star}(\pi_e, D)$ denote the $1 - \delta$ confidence lower bound on $J(\pi_e)$, calculated using weighting scheme $\star$ and CI $*$, where $* \in \{\text{CH}\}$. Let φ denote a safety test such that $\varphi(\pi_e, D, J(\pi_b)) \in \{\texttt{True}, \texttt{False}\}$. Given π_e and data D collected from π_b, φ returns `True` (i.e., the safety test passes) if $L^{*,\star}(\pi_e, D) \geq J(\pi_b)$; otherwise, φ returns `False`. For brevity, we use $L^{*,\star}$ to denote the $1-\delta$ confidence lower bound on $J(\pi_e)$, calculated using any dataset.

3.3 Related Work

This chapter arguably falls under the broad body of work aimed at creating algorithms that can withstand uncertainty introduced at any stage of the RL framework. Some works view a component as an adversary with stochastic behavior. For an overview of risk-averse methods, e.g., those that incorporate stochasticity into the system, refer to the work of Garcia and Fernandez [37].

Other works incorporate an adversary to model worst-case scenarios. In a *model-free* setting, Morimoto and Doya [85] introduced *Robust RL* that models the environment as an adversary to address parameter uncertainty in MDPs. Pinto et al. [94] extend

this work to non-linear policies that are represented as neural networks. Lim et al. [66] consider MDPs with some adversarial state-action pairs. In *model-based* settings, i.e., those that build an explicit model of the system, although an adversary is not present, worst-case analyses assume different components of an MDP are unknown [88, 98, 112]. Using the definition of safety we have introduced, Ghavamzadeh et al. [38] and Laroche et al. [62] created algorithms for learning safe policies in a model-based setting. Although we are also interested in ensuring security for these approaches, we focus on a model-free setting that requires a different set of assumptions and attacker model.

Learning in the presence of an adversary has also been studied in multi-agent RL, where agents have competing goals [68, 93, 100]. Similar to our work, there have been efforts to study the affect of adversarial inputs to algorithms, and create systems that are "robust" to adversarial manipulation, in multi-armed bandits [46, 52, 69, 116, 117] and on image related tasks in deep RL [20, 49, 58]. To our knowledge, there has not been any research on analyzing adversarial attacks on Seldonian RL.

Outside RL, to increase the robustness of supervised and semi-supervised learning, methods have been proposed during training to play adversarial games [32, 42], and to generate adversarial examples [43].

3.4 Problem Formulation

We focus on a worst-case setting, where an attacker modifies training data to maximize the probability that the Seldonian RL algorithm returns an unsafe policy. When the stakes are high—for example, in the application of RL to sepsis treatment in the ICU, wherein training data is generated from hand-written doctors' notes—we do not want to assume that training data contains only minor errors, such as patient height, but also major ones, such as wrong drug or patient name.

Specifically, we will consider the case where the attacker can add k fabricated trajectories to the safety data D. This is related to the case where the attacker can change k of the trajectories in the data—in that setting, the attacker would identify k trajectories to change, and then would replace them with k new ones. After the attacker has identified the k trajectories to change, they are faced with the problem that we solve: which k trajectories to add after the original k were removed.

There are two ways for the safety test to fail: 1) If $J(\pi_e) \geq J(\pi_b)$, the attacker can minimize the estimated performance of π_e by adding k trajectories; and 2) If $J(\pi_e) < J(\pi_b)$, the attacker can maximize the estimated performance of π_e by adding k trajectories. We focus on the latter attacker because the prior case does not cause a poor policy to pass the safety test, but only prevents the identification of a good policy. Therefore, we only consider an attacker who is interested in enabling a worse policy than the behavior policy to pass the safety test. The attacker has the same knowledge inputted to φ, which consists of: 1) The behavior policy, π_b, that generated D; 2) $J(\pi_b)$; 3) The evaluation policy, π_e; 4) The CI, $*$, used by φ; 5) The method to compute importance weights, $\star$; 6) The confidence level, δ; 7) The dataset, D.

Let $\mathcal{D}_n^{\pi_b} = \{\mathcal{D} : \mathcal{D} \subseteq \mathcal{H}_{\pi_b} \text{ and } |\mathcal{D}| = n\}$, i.e., the set of all possible datasets of size n, created by running π_b. For all $k \in \mathbb{Z}^+$, let $m : \mathcal{D}_n^{\pi_b} \times \mathbb{Z}^+ \to \mathcal{D}_{n+k}^{\pi_b}$ be the *attack function*, which indicates a strategy that an attacker might use when appending fabricated trajectories to the dataset. That is, for $D \in \mathcal{D}_n^{\pi_b}$, $m(D, k) \in \mathcal{D}_{n+k}^{\pi_b}$ is the dataset created by using attack strategy m to append k trajectories to D with size n. Let $\mathcal{M}$ denote the set of all possible attack functions m. For notational completeness, we note that m is actually a function of $\mathcal{D}_n^{\pi_b}$, Z^+ and items 1–6 enumerated in the previous paragraph. However, we omit these items for brevity.

Next, we define α-security, which provides one way of quantifying how robust a safety test (and thus a Seldonian algorithm relying on a safety test) is to adversarial perturbations of data in terms of a parameter, $\alpha \geq 0$, such that smaller values of

α correspond to more robustness. We use the following assumptions when defining α-security:

Assumption 1 (**Inferior π_e**). $J(\pi_e) < J(\pi_b)$.

Assumption 2 (**Absolute continuity**). $\forall a \in \mathcal{A}, \forall s \in \mathcal{S}$, $\big(\pi_b(s,a) = 0\big) \implies \big(\pi_e(s,a) = 0\big)$.

Assumption 3 (**φ safety**). *We only consider safety tests that ensure Eq.* (3.1) *is satisfied by any algorithm that returns π_e if $\varphi\big(\pi_e, D, J(\pi_b)\big) = \texttt{True}$, and π_b otherwise. That is, given Assumption 1, φ must satisfy* $\Pr(\varphi(\pi_e, D, J(\pi_b)) = \texttt{True}) < \delta$.

Recall that the attacker is interested in enabling a worse policy than π_b to pass the safety test by appending trajectories to D. In order to succeed, they must manipulate the metric used for decision making by the safety test, i.e., artificially increase $L^{*,\star}$, the $1-\delta$ confidence lower bound on $J(\pi_e)$.

Our definition ensures Eq. (3.1) is satisfied even if D has been corrupted. In fact, an undesirable policy passing the safety test with probability at most δ, must hold across all attack functions, which includes the best attack strategy—one that causes the largest increase in $L^{*,\star}$. This artificial increase in $L^{*,\star}$, due to the best attack strategy, is represented by α, which we write as a function of n, k, π_b, π_e and δ.

Definition 1 (**α-security**). *Under Assumptions 1, 2 and 3, φ is secure with constant α for π_e, π_b, k, and D collected from π_b, where $|D| = n$, if and only if,*

$$\forall m \in \mathcal{M}, \Pr\Big(\varphi\big(\pi_e, m(D,k), J(\pi_b) + \alpha\big) = \texttt{True}\Big) < \delta. \tag{3.2}$$

If a safety test does not satisfy Assumption 3, we can not easily analyze its α-security. For example, any φ using WIS variants does not satisfy Assumption 3 because many CIs often rely on the independence of samples to guarantee probabilistic bounds on their mean. WIS creates dependence between samples by normalizing the

importance weights. However, WIS variants are more likely to be used by safety tests because they work better in practice and require less data than IS. Therefore, to include WIS in our analysis, we define quasi-α-security to be the following.

Definition 2 (**Quasi-α-security**). *Under Assumptions 1 and 2, φ is quasi-secure with constant α for π_e, π_b, k, and D collected from π_b, where $|D| = n$, if and only if,*

$$\forall m \in \mathcal{M}, \Pr\Big(\varphi\big(\pi_e, m(D,k), J(\pi_b) + \alpha\big) = \texttt{True}\Big) \leq \Pr\Big(\varphi\big(\pi_e, D, J(\pi_b)\big) = \texttt{True}\Big). \tag{3.3}$$

Note that Eq. (3.3) implies α-security if φ is "safe". If it is not (as in WIS variants), then φ can be quasi-α-secure, which still gives us a way to measure its robustness to perturbations/attacks on data.

Notice that our definition does not consider how π_e is chosen because the violation of safety comes primarily from the safety test. In addition to the attacker model, our worst-case analysis also stems from this definition of security, which assumes that π_e has lower performance than π_b, but does not quantify how often this occurs. Attacking the data used to select π_e can increase this frequency, but would not change the definition of α-security.

3.5 Analysis of Existing Algorithms

In this section, we present our main contribution which quantifies the α-security of different off-policy performance estimators. Notice that every safety test is α-secure with $\alpha = \infty$, since the test whether $L^{*,\star} > J(\pi_b) + \infty$ will always return `False`. So, when comparing two estimators, it is not sufficient to compare arbitrary values of α for which they are α-secure. Instead, we define the notion of a *minimum* α, which we refer to as α^*.

Definition 3 (**Tight $\boldsymbol{\alpha}$**). *φ is quasi-secure or secure with tight constant α^* if and only if $\alpha^* = min\{\alpha : \varphi$ is quasi-α-secure or α-secure$\}$.*

The value of α can be interpreted as the largest possible increase in $L^{*,\star}$ when an attacker adds k trajectories to D. To compute α^*, given a random dataset, we first must determine the optimal attack strategy, where this increase is largest. This strategy can be determined using the following realization: $L^{\text{CH, IS}}$ and $L^{\text{CH, WIS}}$ are both increasing functions w.r.t. the IS weight and return. Therefore, for a given k, the optimal attack is to create a trajectory that maximizes the value of the IS weight and return. This strategy incurs the maximum "damage" by the attacker.

Notice that fake trajectories created using this strategy are those that have not been performed in the real MDP environment. However, because the transition and reward functions are not known, a practitioner can not distinguish real and fake trajectories. Rare events are critical to account for in RL, and may look like fake trajectories. Perhaps impossible trajectories can be identified using domain-specific knowledge, but that must be analyzed on a per-domain basis.

Second, computing α^* requires identifying the dataset on which the attack strategy is most effective, i.e., the determination of $D \in \mathcal{D}_n^{\pi_b}$ that yields the greatest increase in $L^{*,\star}$. However, this is not possible since the distribution of trajectories for a given π_e is unknown, and thus, $\mathcal{D}_n^{\pi_b}$ is unknown.

Instead, we propose the use of a different value, α', which may be slightly loose relative to the tight α^*, but which we expect captures the relative robustness of the methods that we compare. Instead of $D \in \mathcal{D}_n^{\pi_b}$, $D \in \mathcal{D}_n^{\mathcal{H}}$ is selected, on which the attacker executes the optimal strategy. In other words, the dataset is chosen out of all datasets of size n created by $\mathcal{H}$, the set of all trajectories that can be created by any policy. In Theorem 5, we present the values of α' for each estimator.

Theorem 5. *φ is quasi-secure or secure with $\alpha \geq \alpha'$, where the values of α' are presented in Table 3.1.*

Proof. In Section 3.8.1, we prove that $L^{\text{CH, IS}}$ and $L^{\text{CH, WIS}}$ are increasing functions w.r.t. the IS weight and return. The largest IS weight is a function of π_b, π_e and τ, and

the largest return is always 1. Let any off-policy performance estimator, $L^{*,\star}(\pi_e, D)$, be written as a function f that incorporates an attacker strategy. In other words, $L^{*,\star}(\pi_e, m(D,k)) = f^{*,\star}(D, w_y, g_y, k)$, where w_y is the IS weight and g_y is the return, computed from the trajectory added by the attacker. That is, $f^{*,\star}(D, w_y, g_y, k)$ is the result of applying CI $*$ and weighting scheme $\star$ to D that includes k copies of H^*, which is the trajectory added by the attacker. H^* has an IS weight of w_y and return of g_y.

The optimal attack strategy causes the largest increase in $L^{*,\star}$ such that $L^{*,\star}(\pi_e, m(D,k)) - L^{*,\star}(\pi_e, D)$ is maximized. In Lemma 5 in Section 3.8.1, we prove that an appropriate setting of α is equal to or greater than the largest increase in $L^{*,\star}$ across all datasets, $D \in \mathcal{D}_n^{\pi_b}$, and all attack functions. By Lemma 1, a safety test using $L^{*,\star}$ is quasi-α-secure or α-secure if $\forall D \in \mathcal{D}_n^{\pi_b}$ and $\forall m \in \mathcal{M}$,

$$\alpha \geq L^{*,\star}\big(\pi_e, m(D,k)\big) - L^{*,\star}(\pi_e, D). \tag{3.4}$$

Let $U = \{u : \exists D \in \mathcal{D}_n^{\pi_b}, \exists m \in \mathcal{M}, u = L^{*,\star}\big(\pi_e, m(D,k)\big) - L^{*,\star}(\pi_e, D)\}$, i.e., permissible values of α^* obtained from each $D \in \mathcal{D}_n^{\pi_b}$. Let $\alpha^* = \max_{D \in \mathcal{D}_n^{\pi_b}} \max_{m \in \mathcal{M}} L^{*,\star}\big(\pi_e, m(D,k)\big) - L^{*,\star}(\pi_e, D)$. For all $u \in U$,

$$\begin{aligned} u \leq & \max_{D \in \mathcal{D}_n^{\pi_b}} \max_{m \in \mathcal{M}} L^{*,\star}\big(\pi_e, m(D,k)\big) - L^{*,\star}(\pi_e, D) \\ = & \max_{D \in \mathcal{D}_n^{\pi_b}} f^{*,\star}(D, i^*, 1, k) - L^{*,\star}(\pi_e, D). \end{aligned} \tag{3.5}$$

Besides the optimal attacker strategy, if the distribution of trajectories for a given π_e was also known, the right-hand side of Eq. (3.5), which is α^*, would be computable. Instead, an upper bound of α^* is

$$\alpha^* < \max_{D \in \mathcal{D}_n^{\mathcal{H}}} f^{*,\star}(D, i^*, 1, k) - L^{*,\star}(\pi_e, D) = \alpha', \tag{3.6}$$

Table 3.1: **α–security of current methods (center); settings for clipping weight, c, for α-security written in terms of a user-specified k and α (right).** The minimum IS weight is denoted by $i^{\min}$.

Estimator	α'	c
CH, IS	$i^*\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right)$	$\frac{\alpha}{\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right)}$
CH, WIS	$\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{ki^*}{(i^{\min}+ki^*)}$	$\frac{i^{\min}\left(\alpha - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)}{k\left(1-\alpha+\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)}$

where $\mathcal{D}_n^{\mathcal{H}} = \{\mathcal{D} : \mathcal{D} \subseteq \mathcal{H} \text{ and } |\mathcal{D}| = n\}$. Substituting the definition of $f^{*,\star}$ and $L^{*,\star}$ into the right-hand side of Eq. (3.6) for various weighting schemes and CIs, we solve for a clean expression for α', presented in Table 3.1. For algebraic details, refer to Section 3.8.2. This upper bound, α', of α^* satisfies Eq. (3.4), implying that any α such that $\alpha \geq \alpha'$ also satisfies Eq. (3.4). □

Recall the following to interpret Table 3.1: 1) The IS weight is a ratio of the probability of observing a trajectory under π_e to that of π_b; 2) We only consider MDPs with finite length, τ. The largest IS weight, i^*, is a function of π_e, π_b and τ. To compute i^*: 1) Through brute search, for a single time step, a state and action pair is selected such that the ratio of its probability under π_e to π_b is maximized; 2) If this ratio is greater than 1, the pair is repeated for the length of the trajectory, exponentially increasing the IS weight. Notice that as $i^{\min} \to 0$, the third term of α' for WIS equals 1. Plus, due to the normalization of IS weights, importance weighted returns are always bounded by 1 for WIS. Therefore, α' for WIS is much smaller than that of IS, and in the worst-case, is slightly over 1. The α' for IS is roughly the same as that of WIS, but scaled by i^* that can be massive.

The conclusion to draw is not that WIS is more robust than IS because α-security does not capture the whole story. The difference in true performance of π_e and π_b, in

addition to how accurate $L^{\mathrm{CH},*}$ estimates the performance of π_e on a given uncorrupted dataset, both contribute to the robustness of the estimators to data anomalies. Yet α-security is useful for deciding whether π_e is worth evaluating for a given estimator. As α' increases, $L^{*,\star}$ can artificially be increased by a greater amount; hence, given the choice of two evaluation policies, a practitioner might pick the policy with lower α'. If $\alpha' = 0$, φ for π_e is highly robust against an adversary.

Although we focus on Chernoff-Hoeffding, our analysis also extends to other bounds such as Azuma [4] and Bernstein [77]. Moreover, *per-decision importance sampling* and *weighted per-decision importance sampling* are equivalent to IS and WIS, respectively, when rewards are at the end of a trajectory. So, our analysis applies to these specific cases as well.

3.6 Panacea: An Algorithm for Safe and Secure Policy Improvement

In this section, we describe our algorithm `Panacea`, named after the Greek goddess of healing, that provides α-security, with a user-specified α, if the number of corrupt trajectories in D is upper bounded. That is, the important additional input to the algorithm is the number of adversarial trajectories. The algorithm also takes as input all the information that the attacker already knows; except, is agnostic to *which* or *how many* of the trajectories in $m(D, k)$ have been corrupted.

`Panacea` caps the importance weights using some clipping weight, c. First introduced for RL by Bottou et al. [14], clipping works by computing all the IS weights in D and capping the weights to c, i.e., if any IS weight is greater than c, it is set to c. Given a user-specified α and k as input, `Panacea` computes c, using the values in Table 3.1. It then creates a clipped version of the dataset, denoted by $\texttt{Panacea}(D, c)$. For pseudocode, refer to Algorithm 2 in Section 3.8.3. In Corollary 1, we show that when k is chosen correctly, our algorithm meets a user-specified level of security against

an attacker, whose optimal strategy does not change even if they know we are using `Panacea`. In Corollary 2, we discuss how the clipping weights are computed.

Corollary 1. *If the user upper bounds k, Panacea is quasi-secure or secure with $\alpha \geq \alpha'$, where α' is user-specified.*

Proof. The behavior of $L^{*,\star}$ does not change with `Panacea`: $L^{*,\star}$ is increasing w.r.t. the IS weight and return, regardless of their range of values. So, the optimal attacker strategy remains the same. Let k denote the number of adversarial trajectories added by the attacker, and k' denote the input provided to `Panacea` by the user. When this value is upper bounded correctly, $k' = k$, and `Panacea` computes a clipping weight for the given estimator, denoted by $c^{*,\star}$, using Table 3.1. By Theorem 5, the security of `Panacea` is

$$\begin{aligned}
&\max_{D\in\mathcal{D}_n^{\mathcal{H}}} \max_{m\in\mathcal{M}} L^{*,\star}\big(\pi_e, \texttt{Panacea}(m(D,k), c^{*,\star})\big) - L^{*,\star}\big(\pi_e, \texttt{Panacea}(D, c^{*,\star})\big) \\
&\quad = \max_{D\in\mathcal{D}_n^{\mathcal{H}}} f^{*,\star}(\texttt{Panacea}(D, c^{*,\star}), c^{*,\star}, 1, k) - L^{*,\star}\big(\pi_e, \texttt{Panacea}(D, c^{*,\star})\big) \qquad (3.7) \\
&\quad \leq \alpha'.
\end{aligned}$$

In Section 3.8.3.1, we solve for a clean expression in Eq. (3.7) by substituting in the definition of $f^{*,\star}$, $L^{*,\star}$ and $c^{*,\star}$ for various weighting schemes and CIs, and then simplifying the expression. □

Corollary 2. *If the user upper bounds k, Panacea is quasi-α-secure or α-secure with the values of c in Table 3.1.*

Proof. If the number of adversarial trajectories in D is upper bounded, rewriting Eq. (3.7) in terms of $c^{*,\star}$, and then solving Eq. (3.7) for a clean expression for $c^{*,\star}$ by substituting α', k and $L^{*,\star}$ with the user-specified inputs, equals the clipping

weights found in Table 3.1 for different estimators. For algebraic details, refer to Section 3.8.3.2. □

`Panacea` is more secure than existing methods if the user correctly determines k and selects a value of α that is less than that of existing methods. Only then is $c < i^*$, and `Panacea` clips the k largest IS weights in $m(D, k)$ to c. Additionally, if $c < I_n$, where I_n is the largest IS weight in the uncorrupted dataset D, then some values in D are also clipped. Table 3.2 in Section 3.8.3 is the same as the middle column of Table 3.1 with one modification: i^* are replaced with c. As c decreases, more values are collapsed, $L^{\mathrm{CH},\star}$ is lower, and the probability of returning π_b is higher.

3.7 Empirical Evaluation

We quantify the α-security of two safety tests, with and without `Panacea`, applied to two domains: a grid-world, where the deployment of an unsafe policy has low-stakes, and a diabetes treatment simulation, where the deployment of an unsafe policy could lead to very dangerous outcomes.

3.7.1 Experimental Methodology and Application Domains

Grid-world. In a classic 3×3 grid-world, the agent's goal is to reach the bottom-right corner of the grid, starting from the top-left corner. When viewed as an MDP, actions correspond to directions the agent can move, and states correspond to its current location on the grid. The agent receives a reward of 1, discounted by γ^t, if they reach the bottom-right corner; otherwise all rewards are 0.

Diabetes Treatment Simulation. For the diabetes treatment simulation, we use a Python implementation [115] of an FDA-approved type 1 diabetes Mellitus simulator (T1DMS) by Kovatchev et al. [59] and Man et al. [74]. The simulator simulates the metabolism of a patient with type 1 diabetes, where the body does not make enough insulin, a hormone needed for moving glucose into cells. Insulin

pumps have a bolus calculator that determines how much insulin, specifically known as bolus insulin, must be injected into the body before having a meal. One type of bolus calculator is parameterized by two real-valued parameters, CR and CF. The MDP view of the simulator represents the bolus calculator as a policy, injection doses as actions, and states as the patient's body's reactions to consuming meals and getting insulin injections. We adopt a similar reward function used in previous work that penalizes any deviation from optimum levels of blood glucose [5].

Data Collection. We use RL to search the space of policies for grid-world, and the space of probability distributions over policies for the diabetes domain [108]. For the latter case, we assign the mode of a triangular distribution to a value sampled uniformly from range $[5, 50]$ for CR and $[1, 31]$ for CF—admissible ranges for any diabetic patient. Also, both triangular distributions have the same range from which CR and CF values are sampled. The two modes parameterize the policy space over which we sample policies. D is created by sampling a CR and CF pair from their respective distributions, and observing the return. This reformulation is a bandit problem, where the action corresponds to picking the modes of two triangular distributions from which to sample CR and CF.

Let $\pi(v, u, \theta_1, \theta_2)$ denote a policy representing the joint probability of sampling v under a triangular distribution with mode θ_1 and range $[5, 50]$, and sampling u under a triangular distribution with mode θ_2 and range $[1, 31]$. Instead of finding a trajectory with the largest IS weight and return, the optimal attack strategy is selecting a CR$'$ and CF$'$ such that the ratio of their joint probability under π_e to that of π_b is maximized, i.e., $\arg\max_{\text{CR}' \in [5,50]} \arg\max_{\text{CF}' \in [1,31]} \pi_e(\text{CR}', \text{CF}', \theta_3, \theta_4) / \pi_b(\text{CR}', \text{CF}', \theta_1, \theta_2)$. The attacker then adds k copies of CR$'$ and CF$'$, along with a return of 1, to D. Our results show that corrupting D collected from adult#003 within T1DMS can cause a Seldonian RL algorithm to select a bad policy, i.e., a new distribution over policies with lower return than π_b.

3.7.2 Results and Discussion

We evaluated the number of trajectories that must to be added to D before φ incorrectly returns unsafe policies with probability more than δ. The goal is not to show how much data it takes to "break" each method—the goal is to show that, without `Panacea`, both methods are extremely fragile. For our experimental setup, we selected two policies per domain. We estimated $J(\pi_b) \approx 0.797$ and $J(\pi_e) \approx 0.728$ for grid-world, and $J(\pi_b) \approx 0.218$ and $J(\pi_e) \approx 0.145$ for the diabetes domain, by averaging returns obtained from running each policy 10,000 times. We added k adversarial trajectories based on the optimal attacker strategy to a randomly created D of size 1,500. We executed the safety test by comparing $J(\pi_b)$ to $L^{\text{CH},\star}$, computed using the corrupt data. Figure 3.1 shows the average $L^{\text{CH},\star}$ over 750 trials, as k increases. The error bars for variance are so small that they are negligible. Although not shown, the average probability of passing the safety test across all trials is around 0% before and 100% after the blue and red lines cross the black dotted line, representing $J(\pi_b)$.

The results without `Panacea` show that $L^{\text{CH, IS}}$ and $L^{\text{CH, WIS}}$ cross the black dotted line at $k = 49$ and $k = 1$, respectively, for both domains. For WIS, it only takes a single trajectory for φ to return an unsafe policy; for IS, 49 trajectories only constitute 3.2% of D, where $|D| = 1{,}549$. This could correspond to a morning's worth of data collected from an incorrectly parameterized insulin pump, or the temporary period over which the sensors of a self-driving car are not working due to severe weather conditions. Notice that these results are not indicative of IS being less sensitive than WIS: Because WIS estimates performance more accurately, small changes in its estimate cause the incorrect conclusion that $J(\pi_e) > J(\pi_b)$.

For `Panacea`, we computed the clipping weights, found in Table 3.1 per estimator, using the user-specified α and the actual number of adversarial trajectories added by the attacker. Our method never crosses the black dotted line for either domain at $\alpha = 0.1$ with IS; and it requires 65 adversarial trajectories to break rather than

a single trajectory for the diabetes domain at $\alpha = 0.01$ with WIS. Notice how the parameter settings affect α-security: 1) As α increases, so does the clipping weight. 2) As k increases, the clipping weight decreases to counteract the artificial increase of $L^{\text{CH},\star}$ due to corrupt trajectories. In the worst-case—when the user inputs an α that is greater than the α' of existing methods—`Panacea` requires the same number of adversarial trajectories to break as if not used because D is not clipped at all.

One of the practical considerations when deploying `Panacea` is estimating the number of corrupt trajectories in training data. For areas such as natural language processing and computer vision, a user might address this concern by using known error rates in the data processing pipelines of well-known models. Also, choosing a meaningful value for α might be challenging. Domain-specific knowledge of the range of performance for different policies—especially the difference in performance between good and bad policies—can be useful. But notice that WIS estimates are always bounded by 1, and in practice, IS estimates are even smaller—when computing the IS weight, the probability of a trajectory under π_e is often much smaller than that of π_b. The middle column in Table 3.1 is usually ≥ 1, indicating that standard methods can be completely broken (make pessimal policies appear optimal) easily. The right column shows values of c that make `Panacea` α-secure for *any* $\alpha \in [0, \infty]$. However, plugging in $\alpha \in [0, 1]$ such that $\alpha \leq \alpha'$ gives `Panacea` a meaningful security guarantee.

3.8 Supplementary Proofs

3.8.1 Analysis of Existing Algorithms

Let $f^{*,\star}$ denote a function that incorporates an attacker strategy. When $k = 0$, $f^{\text{CH, IS}}(D, w_y, g_y, k)$ is the result of applying the CH inequality to the IS weighted returns, obtained from D, which additionally includes k copies of a trajectory with an IS weight of w_y and return of g_y. Notice that $f^{*,\star}$ is written in terms of IS weights. The following defines $f^{\text{CH, WIS}}$, written in terms of IS weights, when $k = 0$.

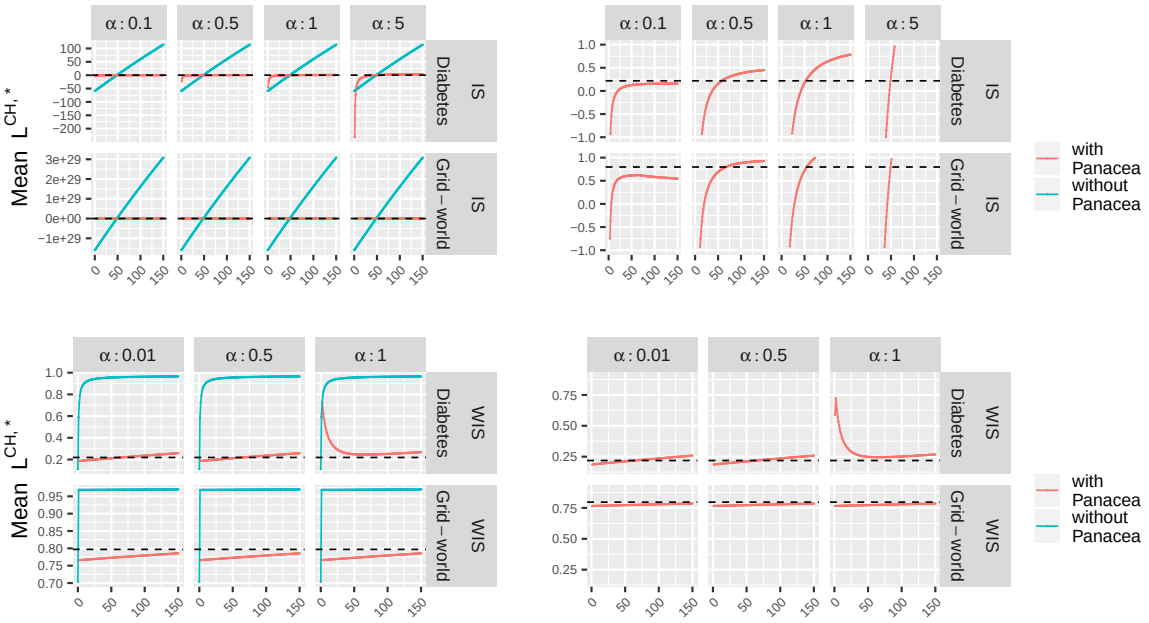

Figure 3.1: **Mean $f^{\mathrm{CH},*}$ as k increases with magnified view (right).** Facets represent the value of α inputted to `Panacea`, and do not matter for existing methods, labelled as "without `Panacea`". For IS, `Panacea` crosses the black dotted line at $k = 59, 53$ and 49 for the diabetes domain, and $k = 68, 55$ and 50 for grid-world, when $\alpha = 0.5, 1$ and 5, respectively. For WIS, `Panacea` crosses the black dotted line at $k = 65, 65$ and 1 for the diabetes domain, when $\alpha = 0.01, 0.5$ and 1, respectively. `Panacea`, using WIS, does not cross the black dotted line for grid-world.

$$f^{\text{CH, WIS}}(D, w_y, g_y, 0) = \frac{1}{\sum_{i=1}^{n} w_i} \sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}}.$$

For the rest of the section, we use the following notation. Let $\mathcal{I} = \{I : \exists a \in \mathcal{A}, \exists s \in \mathcal{S}, I = \prod_{t=0}^{\tau-1} \pi_e(A_t = a, S_t = s)/\pi_b(A_t = a, S_t = s)\}$, i.e., the set of all IS weights that could be obtained from policies π_e and π_b. The maximum and minimum IS weight is denoted by $i^* = \max(\mathcal{I})$ and $i^{\min} = \min(\mathcal{I})$, respectively. For shorthand, let the sum of IS weights in D be written as $\beta = \sum_{i=1}^{n} w_i$. Also, we assume that $\beta > 0$ to ensure that WIS is well-defined.

Next, we define a new term to describe how an attacker can increase the $1 - \delta$ confidence lower bound on the mean of a bounded and real-valued random variable. We say that $f^{*,\star}$ *is adversarially monotonic given its inputs*, if an attacker can maximize $f^{*,\star}$ by maximizing the value of the added samples. For brevity, we say that $f^{*,\star}$ is adversarially monotonic.

Definition 4. *$f^{*,\star}$ is adversarially monotonic for $n > 1$, $k > 0$, π_b, π_e and D if both*

1. *There exists two constants $p \geq 0$ and $q \in [0, 1]$, with $pq \in [0, i^*]$, such that $f^{*,\star}(D, p, q, k) \geq f^{*,\star}(D, p, q, 0)$, i.e., adding k copies of pq does not decrease f;*

2. *$\frac{\partial}{\partial g_y} f^{*,\star}(D, i^*, g_y, k) \geq 0$ and $\frac{\partial}{\partial w_y} f^{*,\star}(D, w_y, 1, k) \geq 0$, with no local maximums, i.e., f is a non-decreasing function w.r.t. the IS weight and return added by the attacker, respectively.*

Definition 4 means that $f^{*,\star}$ is maximized when w_y and g_y is maximized. In other words, the optimal strategy is to add k copies of the trajectory with the maximum IS weight and return. Notice that $f^{*,\star}$ does not incorporate all possible attack functions, $\mathcal{M}$: specifically, the set of attacks, where the attacker can choose to add k different trajectories, is omitted. As described in Theorem 5, to perform a worst-case analysis, only the optimal attack must be incorporated as part of $f^{*,\star}$.

In the following two lemmas, we show that a couple well-known Seldonian algorithms are adversarially monotonic.

Lemma 1. *Under Assumptions 1, 2 and 3, $f^{CH,\ IS}$ is adversarially monotonic.*

Proof. Let $w_y \geq \frac{1}{n}\sum_{i=1}^{n} w_i g_i + \frac{(n+k)}{k}\left(b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right)$ and $g_y = 1$. To show that $w_y g_y \in [0, i^*]$ as stated in (1) in Definition 4, it must be that $w_y \in [0, i^*]$. For all $i \in \{1, \ldots, n\}$, $w_i g_i \in [0, i^*]$. Thus, for any given dataset, $0 \leq 1/n \sum_{i=1}^{n} w_i g_i \leq i^*/n$. Using this fact, for any given D, the range of w_y is

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n}(0) + \frac{(n+k)}{k}\left(b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \leq\ & w_y \\
\leq\ & \frac{1}{n}\sum_{i=1}^{n}(i^*) + \frac{(n+k)}{k}\left(b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \\
\frac{b(n+k)}{k}\Bigg(\underbrace{\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \sqrt{\frac{\ln(1/\delta)}{2n}}}_{<0}\Bigg) \leq\ & w_y \\
\leq\ & \frac{i^*}{n} + \underbrace{\frac{b(n+k)}{k}\left(\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \sqrt{\frac{\ln(1/\delta)}{2n}}\right)}_{<0} \leq i^*.
\end{aligned}$$

Therefore, w_y can be selected such that $w_y g_y \in [0, i^*]$. It follows that

$$\begin{aligned}
f^{\text{CH, IS}}(D, w_y, 1, k) =& \frac{1}{n+k}\sum_{i=1}^{n} w_i g_i + \frac{k}{n+k}(w_y)(1) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
\geq& \frac{1}{n+k}\sum_{i=1}^{n} w_i g_i + \frac{k}{n+k}\left(\frac{1}{n}\sum_{i=1}^{n} w_i g_i + \frac{(n+k)}{k}\left(b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right)\right) \\
& - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
=& \frac{1}{n}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}} \\
=& f^{\text{CH, IS}}(D, w_y, g_y, 0).
\end{aligned}$$

Next, we show that (2) in Definition 4 holds.

$$\begin{aligned}
\frac{\partial}{\partial w_y} f^{*,\star}(D, w_y, g_y, k) =& \frac{\partial}{\partial w_y}\left(\sum_{i=1}^{n} \frac{w_i g_i}{n+k}\right) + \frac{k w_y g_y}{n+k} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
=& \frac{k g_y}{n+k} \\
\frac{\partial}{\partial w_y} f^{*,\star}(D, w_y, 1, k) =& \frac{k}{n+k}. \\
\frac{\partial}{\partial g_y} f^{*,\star}(D, w_y, g_y, k) =& \frac{\partial}{\partial g_y}\left(\sum_{i=1}^{n} \frac{w_i g_i}{n+k}\right) + \frac{k w_y g_y}{n+k} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
=& \frac{k w_y}{n+k} \\
\frac{\partial}{\partial g_y} f^{*,\star}(D, i^*, g_y, k) =& \frac{k i^*}{n+k}.
\end{aligned}$$

Notice that both partial derivatives are non-negative when $g_y = 1$ and $w_y = i^*$, respectively. To find any critical points, the following equations are solved simultaneously: $\partial/\partial g_y f^{\text{CH, WIS}}(D, w_y, g_y, k) = 0$ and $\partial/\partial w_y f^{\text{CH, WIS}}(D, w_y, g_y, k) = 0$. Notice that points along the line $(w_g, 0)$ and $(0, g_y)$ are all critical points. The following partial derivatives are computed to classify these points.

$$\begin{aligned}
\frac{\partial}{\partial (w_y)^2}(D, w_y, g_y, k) =& 0. \\
\frac{\partial}{\partial (g_y)^2}(D, w_y, g_y, k) =& 0. \\
\frac{\partial}{\partial g_y w_y}(D, w_y, g_y, k) =& \frac{k}{n+k}.
\end{aligned}$$

Using the second partial derivative test, the critical points are substituted into the following equation.

$$\frac{\partial}{\partial (w_y)^2} \cdot \frac{\partial}{\partial (g_y)^2} - \left(\frac{\partial}{\partial g_y w_y}\right)^2 = -\left(\frac{k}{n+k}\right)^2,$$

which is less than zero. Therefore, points along the line $(w_g, 0)$ and $(0, g_y)$ are saddle points. □

Lemma 2. *Under Assumptions 1 and 2, $f^{CH,\ WIS}$ is adversarially monotonic.*

Proof. First, we show that (1) in Definition 4 holds with $g_y = 1$ and $w_y = 0$.

$$\begin{aligned}
f^{\text{CH, WIS}}(D, w_y, g_y, k) =& \frac{1}{kw_y + \beta}\left(kw_y g_y + \sum_{i=1}^{n} w_i g_i\right) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
f^{\text{CH, WIS}}(D, 0, 1, k) =& \frac{1}{\beta}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
>& \frac{1}{\beta}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}} \qquad (3.8) \\
=& f^{\text{CH, WIS}}(D, w_y, g_y, 0),
\end{aligned}$$

where Eq. (3.8) follows from $b\sqrt{\frac{\ln(1/\delta)}{2n}} > b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}}$. Second, we show that (2) in Definition 4 holds.

$$\begin{aligned}
\frac{\partial}{\partial w_y} f^{\text{CH, WIS}}(V, w_y, g_y, k) =& -\frac{k\sum_{i=1}^{n} w_i g_i}{\left(kw_y + \beta\right)^2} - \frac{k^2 w_y g_y}{(kw_y + \beta)^2} + \frac{kg_y}{(kw_y + \beta)} \\
=& -\frac{k\sum_{i=1}^{n} w_i g_i}{\left(kw_y + \beta\right)^2} - \frac{k^2 w_y g_y}{(kw_y + \beta)^2} + \frac{kg_y(kw_y + \beta)}{(kw_y + \beta)^2} \\
=& -\frac{k\sum_{i=1}^{n} w_i g_i}{\left(kw_y + \beta\right)^2} + \frac{kg_y\beta}{(kw_y + \beta)^2} \\
=& -\frac{k\sum_{i=1}^{n} w_i g_i}{\left(kw_y + \beta\right)^2} + \frac{k\sum_{i=1}^{n} w_i g_y}{(kw_y + \beta)^2} \\
=& \frac{k}{(\beta + kw_y)^2}\sum_{i=1}^{n} w_i(g_y - g_i) \\
\frac{\partial}{\partial w_y} f^{\text{CH, WIS}}(V, w_y, 1, k) =& \frac{k}{(\beta + kw_y)^2}\sum_{i=1}^{n} w_i(1 - g_i). \qquad (3.9)
\end{aligned}$$

Notice that Eq. (3.9) is non-negative: 1) When $g_y = 1$, Eq. (3.9) is positive as long as there exists at least one $g_i < 1$ for $i \in \{1, \ldots, n\}$; 2) If all $g_i = 1$ in D, then Eq. (3.9) is zero. The following is the derivative of $f^{\text{CH, WIS}}(D, w_y, g_y, k)$ w.r.t. g_y.

$$\frac{\partial}{\partial g_y} f^{\text{CH, WIS}}(D, w_y, g_y, k) = \frac{k w_y}{(\beta + k w_y)} \tag{3.10}$$
$$\frac{\partial}{\partial g_y} f^{\text{CH, WIS}}(D, i^*, g_y, k) = \frac{k i^*}{(\beta + k i^*)},$$

which is also non-negative. To find any critical points, the following equations are solved simultaneously: $\partial/\partial g_y f^{\text{CH, WIS}}(D, w_y, g_y, k) = 0$ and $\partial/\partial w_y f^{\text{CH, WIS}}(D, w_y, g_y, k) = 0$. Notice that Eq. (3.10) is zero when $w_y = 0$. Plugging $w_y = 0$ into $\partial/\partial w_y f^{\text{CH, WIS}}(D, w_y, g_y, k) = 0$, and then solving for g_y, yields the x coordinate of a critical point.

$$\begin{aligned}
\frac{k}{(\beta + k(0))^2} \sum_{i=1}^{n} w_i (g_y - g_i) &= 0 \\
\frac{k}{\beta^2} \sum_{i=1}^{n} w_i (g_y - g_i) &= 0 \\
g_y \sum_{i=1}^{n} w_i - \sum_{i=1}^{n} w_i g_i &= 0 \\
g_y &= \frac{\sum_{i=1}^{n} w_i g_i}{\beta}.
\end{aligned}$$

The following partial derivatives are computed to classify whether $(0, \sum_{i=1}^{n} w_i g_i / \beta)$ is a minimum, maximum or saddle point:

$$\begin{aligned}
\frac{\partial}{\partial (w_y)^2}(D, w_y, g_y, k) &= \frac{-2k^2}{(\beta + k w_y)^3} \sum_{i=1}^{n} w_i (g_y - g_i). \\
\frac{\partial}{\partial (g_y)^2}(D, w_y, g_y, k) &= 0. \\
\frac{\partial}{\partial g_y w_y}(D, w_y, g_y, k) &= \frac{\partial}{\partial w_y} \frac{k w_y}{(\beta + k w_y)} \\
&= \frac{k\beta}{(\beta + k w_y)^2}.
\end{aligned}$$

Using the second partial derivative test, the critical point is substituted into the following equation.

$$\begin{aligned}\frac{\partial}{\partial(w_y)^2}\cdot\frac{\partial}{\partial(g_y)^2}-\left(\frac{\partial}{\partial g_y w_y}\right)^2 &= 0-\left(\frac{k\beta}{(\beta+k(0))^2}\right)^2\\ &= -\left(\frac{k\beta}{\beta^2}\right)^2,\end{aligned}$$

which is less that zero. Therefore, $(w_y = 0, g_y = \sum_{i=1}^{n} w_i g_i/\beta)$ is a saddle point. □

Next, we describe the trajectory that must be added to D to execute the optimal attack.

Definition 5 (**Optimal Attack**). *An optimal attack strategy for $k > 0$ is to select*

$$\arg\max_{H\in\mathcal{H}_{\pi_e}} f^{*,\star}\big(D, w_y = w(H,\pi_e,\pi_b), g_y = g(H), k\big).$$

Definition 6 (**Optimal Trajectory**). *Given that a maximum exists, let $(a', s') \in \arg\max_{a\in\mathcal{A},s\in\mathcal{S}} \frac{\pi_e(a,s)}{\pi_b(a,s)}$. If $\frac{\pi_e(a,s)}{\pi_b(a,s)} > 1$, let $H^* = \{S_0 = s', A_0 = a', R_0 = 1, \ldots, S_{\tau-1} = s', A_{\tau-1} = a', R_{\tau-1} = 1\}$. Otherwise, let $H^* = \{S_0 = s', A_0 = a', R_0 = 1\}$.*

Theorem 6. *For any adversarially monotonic off-policy estimator, the optimal attack strategy is to add k repetitions of H^* to D.*

Proof. An optimal attack strategy is equivalent to

$$\arg\max_{H\in\mathcal{H}_{\pi_e}} f^{*,\star}\big(D, w(H,\pi_e,\pi_b), g(H), k\big) = \arg\max_{i^*\in\mathcal{I}, g^*\in[0,1]} f^{*,\star}\big(D, i^*, g^*, k\big).$$

For any off-policy estimator that is adversarially monotonic, by (1) of Definition 4, there exists a pq such that

$$f^{*,\star}(D,p,q,k) \geq f^{*,\star}(D,p,q,0).$$

A return that maximizes $f^{*,\star}(D, w_y, g_y, k)$ implies that

$$\max_{g^*\in[0,1]} f^{*,\star}(D,p,g^*,k) \geq f^{*,\star}(D,p,q,k).$$

$f^{\text{CH, IS}}$ and $f^{\text{CH, WIS}}$ are non-decreasing w.r.t. the return. Therefore,

$$\arg\max_{g^* \in [0,1]} f^{*,\star}(D, p, g^*, k) = \max_{g^* \in [0,1]} g^*.$$

Setting $g^* = 1$, an importance weight that maximizes $f^{*,\star}(D, w_y, 1, k)$ implies that

$$\max_{i^* \in \mathcal{I}} f^{*,\star}(D, i^*, 1, k) \geq f^{*,\star}(D, p, 1, k).$$

$f^{\text{CH, IS}}$ and $f^{\text{CH, WIS}}$ are also non-decreasing w.r.t. the importance weight. So,

$$\arg\max_{i^* \in \mathcal{I}} f^{*,\star}(D, i^*, 1, k) = \max_{i^* \in \mathcal{I}} i^*.$$

Since the IS weight is a product of ratios over the length of a trajectory, the ratio at a single time step is maximized.

$$\begin{aligned}
\max_{i^* \in \mathcal{I}} i^* &= \max_{a \in \mathcal{A}, s \in \mathcal{S}} \prod_{t=0}^{\tau-1} \frac{\pi_e(A_t = a, S_t = s)}{\pi_b(A_t = a, S_t = s)} \\
&= \begin{cases} \left(\max_{a \in \mathcal{A}, s \in \mathcal{S}} \frac{\pi_e(a,s)}{\pi_b(a,s)} \right)^{\tau} & \text{if } \max_{a \in \mathcal{A}, s \in \mathcal{S}} \frac{\pi_e(a,s)}{\pi_b(a,s)} > 1, \\ \max_{a \in \mathcal{A}, s \in \mathcal{S}} \frac{\pi_e(a,s)}{\pi_b(a,s)} & \text{otherwise.} \end{cases}
\end{aligned}$$

To create H^*, if the ratio at a single time step is greater than 1, a' and s' is repeated for the maximum length of the trajectory, τ; otherwise, a' and s' is repeated only for a single time step. Thus, H^* represents the trajectory with the largest return and importance weight. □

Next, we show how Eq. (3.3) and Eq. (3.2), that define quasi-α-security and α-security, respectively, apply to $L^{*,\star}$. Specifically, we show that a safety test using $L^{*,\star}$ as a metric is a valid safety test that first predicts the performance of π_e using D, and then bounds the predicted performance with high probability. If $L^{*,\star}(\pi_e, D) > J(\pi_b)$, the safety test returns `True`; otherwise it returns `False`.

Lemma 3. *A safety test using $L^{\star,\star}$ is quasi-α-secure if* $\forall m \in \mathcal{M}, \Pr\Big(L^{\star,*}\big(\pi_e, m(D,k)\big) > J(\pi_b) + \alpha\Big) \leq \Pr\Big(L^{\star,*}\big(\pi_e, D\big) > J(\pi_b)\Big)$.

Proof. For $x \in \mathbb{N}^+$, let $\mathcal{P} : \Pi \times D_n^{\pi_b} \to \mathbb{R}^x$ denote any function to predict the performance of some $\pi_e \in \Pi$, using data D collected from π_b. Also, let $\mathcal{B} : \mathbb{R}^x \times [0,1] \to \mathbb{R}$ denote any function that bounds performance with high probability, $1-\delta$, where $\delta \in [0,1]$. Starting with the definition of quasi-α-security, we have that $\forall m \in \mathcal{M}$,

$$\Pr\Big(\varphi\big(\pi_e, m(D,k), J(\pi_b)+\alpha\big) = \texttt{True}\Big) \leq \Pr\Big(\varphi\big(\pi_e, D, J(\pi_b)\big) = \texttt{True}\Big)$$

$$\iff \Pr\Big(\mathcal{B}\big(\mathcal{P}(\pi_e, m(D,k)), \delta\big) > J(\pi_b)+\alpha\Big) \leq \Pr\Big(\mathcal{B}\big(\mathcal{P}(\pi_e, D), \delta\big) > J(\pi_b)\Big)$$

$$\iff \Pr\Big(L^{\star,*}\big(\pi_e, m(D,k)\big) > J(\pi_b)+\alpha\Big) \leq \Pr\Big(L^{\star,*}\big(\pi_e, D\big) > J(\pi_b)\Big).$$

□

Lemma 4. *A safety test using $L^{\star,\star}$ is α-secure if* $\forall m \in \mathcal{M}, \Pr\Big(L^{\star,*}\big(\pi_e, m(D,k)\big) > J(\pi_b) + \alpha\Big) < \delta$.

Proof. For $x \in \mathbb{N}^+$, let $\mathcal{P} : \Pi \times D_n^{\pi_b} \to \mathbb{R}^x$ denote any function to predict the performance of some $\pi_e \in \Pi$, using data D collected from π_b. Also, let $\mathcal{B} : \mathbb{R}^x \times [0,1] \to \mathbb{R}$ denote any function that bounds performance with high probability, $1-\delta$, where $\delta \in [0,1]$. Starting with the definition of α-security, we have that $\forall m \in \mathcal{M}$,

$$\Pr\Big(\varphi\big(\pi_e, m(D,k), J(\pi_b)+\alpha\big) = \texttt{True}\Big) < \delta$$

$$\iff \Pr\Big(\mathcal{B}\big(\mathcal{P}(\pi_e, m(D,k)), \delta\big) > J(\pi_b)+\alpha\Big) < \delta$$

$$\iff \Pr\Big(L^{\star,*}\big(\pi_e, m(D,k)\big) > J(\pi_b)+\alpha\Big) < \delta.$$

□

In Lemma 5, we describe a condition that must hold in order to compute a valid α. The condition states that a valid α must be equal to or greater than the largest increase in the $1-\delta$ confidence lower bound on $J(\pi_e)$ across all datasets $D \in \mathcal{D}_n^{\pi_b}$ and all attack strategies (i.e., the optimal attack).

Lemma 5. *A safety test using $L^{*,\star}$ is quasi-α-secure or α-secure if $\forall D \in \mathcal{D}_n^{\pi_b}$ and $\forall m \in \mathcal{M}$, $L^{*,\star}\big(\pi_e, m(D,k)\big) \leq L^{*,\star}(\pi_e, D) + \alpha$.*

Proof. If $L^{*,\star}\big(\pi_e, m(D,k)\big) \leq L^{*,\star}(\pi_e, D) + \alpha$, then

$$L^{*,\star}(\pi_e, D) \geq L^{*,\star}\big(\pi_e, m(D,k)\big) - \alpha. \tag{3.11}$$

A safety test checks whether $L^{*,\star}(\pi_e, D) > J(\pi_b)$. When Eq. (3.11) holds $\forall D \in \mathcal{D}_n^{\pi_b}$ and $\forall m \in \mathcal{M}$,

$$\Pr(L^{*,\star}(\pi_e, D) > J(\pi_b)) \geq \Pr(L^{*,\star}\big(\pi_e, m(D,k)\big) - \alpha > J(\pi_b)), \tag{3.12}$$

and hence via algebra that

$$\Pr(L^{*,\star}\big(\pi_e, m(D,k)\big) > J(\pi_b) + \alpha) \leq \Pr(L^{*,\star}(\pi_e, D) > J(\pi_b)),$$

which, by Lemma (3), implies that a safety test using $L^{*,\star}$ is quasi-α-secure. In the case of α-security, by Assumption 3, we require a "safe" safety test. That is,

$$\Pr(L^{*,\star}(\pi_e, D) > J(\pi_b)) < \delta. \tag{3.13}$$

From the transitive property of $\geq$, we can conclude from Eq. (3.12) and Eq. (3.13) that

$$\Pr(L^{*,\star}\big(\pi_e, m(D,k)\big) - \alpha > J(\pi_b)) < \delta,$$

and hence via algebra that

$$\Pr(L^{*,\star}\big(\pi_e, m(D,k)\big) > J(\pi_b) + \alpha) < \delta,$$

which, by Lemma (4), implies that a safety test using $L^{*,\star}$ is α-secure. □

3.8.2 Proof of Theorem 5

Equation (3.6) for the estimator that uses CH and IS is the following.

$$\begin{aligned}
\alpha' &= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, IS}}(D, i^*, 1, k) - L^{\text{CH, IS}}(\pi_e, D) \\
&= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} \frac{1}{n+k} \sum_{i=1}^{n} w_i g_i + \frac{k}{n+k}(i^*)(1) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \left(\frac{1}{n}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \\
&= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} b\sqrt{\frac{\ln(1/\delta)}{2n}} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\Big(i^* - \frac{\sum_{i=1}^{n} w_i g_i}{n}\Big).
\end{aligned}$$

Recall that b represents the upper bound of all IS weighted returns. Let $b = i^*$, and $g_i = 0$ for all $i \in \{1, \ldots, n\}$.

$$\begin{aligned}
\alpha' =& i^*\sqrt{\frac{\ln(1/\delta)}{2n}} - i^*\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}(i^* - 0) \\
=& i^*\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right).
\end{aligned}$$

Equation (3.6) for the estimator that uses CH and WIS is the following.

$$\begin{aligned}
\alpha' &= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, WIS}}(D, i^*, 1, k) - L^{\text{CH, WIS}}(\pi_e, D) \\
&= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} \frac{1}{ki^* + \sum_{i=1}^{n} w_i}\left(\sum_{i=1}^{n} w_i g_i + k(i^*)(1)\right) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \left(\frac{1}{\sum_{i=1}^{n} w_i}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \\
&= \max_{D \in \mathcal{D}_n^{\mathcal{H}}} b\sqrt{\frac{\ln(1/\delta)}{2n}} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{ki^*}{(ki^* + \beta)}\left(1 - \frac{\sum_{i=1}^{n} w_i g_i}{\beta}\right).
\end{aligned}$$

Let $g_i = 0$ for all $i \in \{1, \ldots, n\}$. Also, notice that $b = 1$ because importance weighted returns are in range $[0, 1]$ for WIS.

$$\alpha' = \max_{D \in \mathcal{D}_n^{\mathcal{H}}} \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{ki^*}{(ki^* + \beta)}.$$

Recall that $\beta \neq 0$. So, let $w_i = 0$ for all $i \in \{1, \ldots, n-1\}$ and $w_n = i^{\min}$.

$$\alpha' = \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{ki^*}{(i^{\min} + ki^*)}.$$

3.8.3 Panacea: An Algorithm for Safe and Secure Policy Improvement

Table 3.2: $\boldsymbol{\alpha}$**–security of** `Panacea`**.**

Estimator	$\boldsymbol{\alpha}$
CH, IS	$c\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right)$
CH, WIS	$\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{kc}{(i^{\min}+kc)}$

Algorithm 2: `Panacea`(D, π_e, α, k)

```
Compute c, using α and k, given estimator
for H ∈ D do
    if IS weight computed using H is greater than c then
        Set IS weight to c
    end if
end for
return clipped D
```

Algorithm 2. This algorithm ensures a user-specified level of α-security when input k is selected such that it represents the correct number of trajectories added by an adversary.

3.8.3.1 Proof of Corollary 1

Let α' and k' denote the user-specified inputs to `Panacea`. Based on Table 3.1, $c^{\text{CH, IS}} = {}^{\alpha'}/\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right)$ if $k' = k$. Recall that b is the upper bound on all

IS weighted returns. Due to clipping, $b = c^{\text{CH, IS}}$, and let $g_i = 0$ for all $i \in \{1, \ldots, n\}$. The result of Eq. (3.7) for the estimator that uses CH and IS is the following:

$$
\begin{aligned}
&\max_{D \in \mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, IS}}(\texttt{Panacea}(D, c^{\text{CH, IS}}), c^{\text{CH, IS}}, 1, k) - L^{\text{CH, IS}}\big(\pi_e, \texttt{Panacea}(D, c^{\text{CH, IS}})\big) \\
&\quad = \max_{D \in \mathcal{D}_n^{\mathcal{H}}} \frac{1}{n+k} \sum_{i=1}^{n} w_i g_i + \frac{k}{n+k}(c^{\text{CH, IS}})(1) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \left(\frac{1}{n}\sum_{i=1}^{n} w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \\
&\quad = \max_{D \in \mathcal{D}_n^{\mathcal{H}}} b\sqrt{\frac{\ln(1/\delta)}{2n}} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\Big(c^{\text{CH, IS}} - \frac{\sum_{i=1}^{n} w_i g_i}{n}\Big) \\
&\quad = c^{\text{CH, IS}}\sqrt{\frac{\ln(1/\delta)}{2n}} - c^{\text{CH, IS}}\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}(c^{\text{CH, IS}} - 0) \\
&\quad = c^{\text{CH, IS}}\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right) \\
&\quad = \frac{\alpha'}{\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}} \cdot \left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right) \\
&\quad = \alpha'.
\end{aligned}
$$

For WIS, recall that no matter how the clipping weight is set, $b \leq 1$ because importance weighted returns are in range $[0, 1]$, and $\beta \neq 0$. So, let $w_i = 0$ for all $i \in \{1, \ldots, n-1\}$ and $w_n = i^{\min}$. Also, let $g_i = 0$ for all $i \in \{1, \ldots, n\}$. Based on Table 3.1, $c^{\text{CH, WIS}} = i^{\min}\Big(\alpha' - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\Big)/k\Big(1 - \alpha' + \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\Big)$ if $k' = k$. Equation (3.7) for the estimator that uses CH and WIS is the following.

$$\max_{D\in\mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, WIS}}(\texttt{Panacea}(D, c^{\text{CH, WIS}}), c^{\text{CH, WIS}}, 1, k) - L^{\text{CH, WIS}}\big(\pi_e, \texttt{Panacea}(D, c^{\text{CH, WIS}})\big)$$

$$\begin{aligned}
&= \max_{D\in\mathcal{D}_n^{\mathcal{H}}} \frac{1}{kc^{\text{CH, WIS}} + \sum_{i=1}^n w_i}\left(\sum_{i=1}^n w_i g_i + k(c^{\text{CH, WIS}})(1)\right) - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} - \\
&\quad \left(\frac{1}{\sum_{i=1}^n w_i}\sum_{i=1}^n w_i g_i - b\sqrt{\frac{\ln(1/\delta)}{2n}}\right) \\
&= \max_{D\in\mathcal{D}_n^{\mathcal{H}}} b\sqrt{\frac{\ln(1/\delta)}{2n}} - b\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{kc^{\text{CH, WIS}}}{(kc^{\text{CH, WIS}} + \beta)}\left(1 - \frac{\sum_{i=1}^n w_i g_i}{\beta}\right) \\
&\leq \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{kc^{\text{CH, WIS}}}{(kc^{\text{CH, WIS}} + i^{\min})} \\
&= \left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right) + \frac{\left(\alpha' - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)}{\left(\alpha' - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right) + \left(1 - \alpha' + \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)} \\
&= \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \alpha' - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
&= \alpha'.
\end{aligned}$$

3.8.3.2 Proof of Corollary 2

Let α and k' denote the user-specified inputs to `Panacea`. If $k' = k$, i.e., the user inputs the correct number of trajectories added by the attacker, the result of Eq. (3.7) for the estimator that uses CH and IS is the following.

$$\begin{aligned}
\alpha &= \max_{D\in\mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, IS}}(\texttt{Panacea}(D, c), c, 1, k) - L^{\text{CH, IS}}\big(\pi_e, \texttt{Panacea}(D, c)\big) \\
\alpha &= c\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right) \\
c &= \frac{\alpha}{\left(\sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{k}{(n+k)}\right)}.
\end{aligned}$$

If $k' = k$, the result of Eq. (3.7) for the estimator that uses CH and WIS is the following.

$$\max_{D \in \mathcal{D}_n^{\mathcal{H}}} f^{\text{CH, WIS}}(\texttt{Panacea}(D, c), c, 1, k) - L^{\text{CH, WIS}}\big(\pi_e, \texttt{Panacea}(D, c)\big) \leq \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{kc}{(kc + i^{\min})}. \tag{3.14}$$

Setting the right-hand side of Eq. (3.14) to α, and solving for c equals

$$\begin{aligned}
\alpha =& \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} + \frac{kc}{(i^{\min} + kc)} \\
\frac{kc}{(i^{\min} + kc)} =& \alpha - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
kc - kc\alpha + kc\sqrt{\frac{\ln(1/\delta)}{2n}} - kc\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} =& i^{\min}\alpha - i^{\min}\sqrt{\frac{\ln(1/\delta)}{2n}} + i^{\min}\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
kc\left(1 - \alpha + \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right) =& i^{\min}\alpha - i^{\min}\sqrt{\frac{\ln(1/\delta)}{2n}} + i^{\min}\sqrt{\frac{\ln(1/\delta)}{2(n+k)}} \\
c =& \frac{i^{\min}\left(\alpha - \sqrt{\frac{\ln(1/\delta)}{2n}} + \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)}{k\left(1 - \alpha + \sqrt{\frac{\ln(1/\delta)}{2n}} - \sqrt{\frac{\ln(1/\delta)}{2(n+k)}}\right)}.
\end{aligned}$$

CHAPTER 4

CONCLUSIONS

In this dissertation, we provided two instances demonstrating the importance of and need for randomness in computer science. It is due to the random nature of probabilistic data structures and the data generation process that we are able to utilize concentration inequalities, which are powerful tools. The application of these tools help conserve bandwidth while ensuring a desired decode rate in Chapter 2, and guarantee safety in Chapter 3.

In Chapter 2, we introduced a novel solution to the problem of determining a subset of items from a larger set two parties hold in common, using a novel combination of two probabilistic data structures, Bloom filters and IBLTs. We also provided a solution to the more general case, where one party is missing some or all of the items. Specifically, we described how to parameterize the probabilistic data structures, using the Chernoff bound, in order to meet a desired decode rate. Through a detailed evaluation using simulations and real-world deployment, we compared our method to existing systems, showing that it requires less data transmission over a network and is more resilient to attack than previous approaches.

In Chapter 3, we analyzed a couple of safety tests that use the Chernoff-Hoeffding bound. More specifically, we presented a new measure, called α-security, to quantify the susceptibility of these safety tests to data corruption attacks, which represent the insertion of non-random samples to a dataset. The effects of such attacks can be dire, breaking the safety guarantee provided by these methods, and consequently, highlighting the importance of collecting data randomly. Recognizing that anomalies

and disturbances can hinder this collection, we also introduced a new algorithm, `Panacea`, which guarantees a user-specified level of security even when data is not collected completely randomly.

BIBLIOGRAPHY

[1] Yaser S Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin. *Learning from data*, volume 4. AMLBook New York, NY, USA:, 2012.

[2] Daniel Albright, Arrick Lanfranchi, Anwen Fredriksen, William F. Styler IV, Colin Warner, Jena D. Hwang, Jinho D. Choi, Dmitriy Dligach, Rodney D. Nielsen, James Martin, et al. Towards comprehensive syntactic and semantic annotations of the clinical narrative. *Journal of the American Medical Informatics Association*, 20(5):922–930, 2013.

[3] Jean-Philippe Aumasson and Daniel J. Bernstein. SipHash: A Fast Short-Input PRF. In *Proc. Progress in Cryptology (INDOCRYPT)*, pages 489–508, December 2012.

[4] Kazuoki Azuma. Weighted sums of certain dependent random variables. *Tohoku Mathematical Journal, Second Series*, 19(3):357–367, 1967.

[5] Meysam Bastani. Model-free intelligent diabetes management using machine learning. *M.S. Thesis, University of Alberta*, 2014.

[6] Jon Louis Bentley and Andrew Chi-Chih Yao. An almost optimal algorithm for unbounded searching (see also `https://en.wikipedia.org/wiki/Exponential_search`). *Information Processing Letters.*, 5(3):82–87, Aug 1976. doi: 10.1016/0020-0190(76)90071-5.

[7] George Bissias. Graphene Pull Request. `https://github.com/BitcoinUnlimited/BitcoinUnlimited/pull/973`, July 2018.

[8] George Bissias. An Algorithm for Bounding the Probability of r-core Formation in k-uniform Random Hypergraphs. arXiv preprint, Feb 1 2019. URL `http://arxiv.org/abs/1901.04934`.

[9] George Bissias and Brian Levine. BUIP093: Graphene Relay. `https://github.com/BitcoinUnlimited/BUIP/blob/master/093.mediawiki`, July 26 2018.

[10] Bitcoin ABC. The bitcoin abc vision. `https://medium.com/@Bitcoin_ABC/the-bitcoin-abc-vision-f7f87755979f`, August 24 2018.

[11] Bitcoin Unlimited. Bitcoin Cash Development And Testing Accord: Bitcoin Unlimited Statement. `https://www.bitcoinunlimited.info/cash-development-plan`, 2018.

[12] Burton H. Bloom. Space/Time Trade-offs in Hash Coding with Allowable Errors. *Commun. ACM*, 13(7):422–426, July 1970.

[13] Anudhyan Boral and Michael Mitzenmacher. Multi-party set reconciliation using characteristic polynomials. In *Proc. Annual Allerton Conference on Communication, Control, and Computing*, pages 1182–1187, October 2014.

[14] Léon Bottou, Jonas Peters, Joaquin Quiñonero-Candela, Denis X Charles, D Max Chickering, Elon Portugaly, Dipankar Ray, Patrice Simard, and Ed Snelson. Counterfactual reasoning and learning systems: The example of computational advertising. *The Journal of Machine Learning Research*, 14(1):3207–3260, 2013.

[15] Stéphane Boucheron, Gábor Lugosi, and Pascal Massart. *Concentration inequalities: A nonasymptotic theory of independence*. Oxford university press, 2013.

[16] Andrei Broder and Michael Mitzenmacher. Network applications of bloom filters: A survey. *Internet mathematics*, 1(4):485–509, 2004.

[17] Andrej Brodnik and J Ian Munro. Membership in constant time and almost-minimum space. *SIAM Journal on Computing*, 28(5):1627–1640, 1999.

[18] Vitalik Buterin and Virgil Griffith. Casper the Friendly Finality Gadget. `https://arxiv.org/abs/1710.09437`, Oct 2017.

[19] John W Byers, Jeffrey Considine, Michael Mitzenmacher, and Stanislav Rost. Informed content delivery across adaptive overlay networks. *IEEE/ACM transactions on networking*, 12(5):767–780, 2004.

[20] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[21] Larry Carter, Robert Floyd, John Gill, George Markowsky, and Mark Wegman. Exact and approximate membership testers. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, pages 59–65. ACM, 1978. URL `http://doi.acm.org/10.1145/800133.804332`.

[22] Amit Chakrabarti. Data stream algorithms lecture notes. 2020.

[23] Herman Chernoff et al. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *The Annals of Mathematical Statistics*, 23(4):493–507, 1952.

[24] Matt Corallo. Bip152: Compact block relay. `https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki`, April 2016.

[25] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On Scaling Decentralized Blockchains. In *Proc. Financial Cryptography and Data Security*, pages 106–125, February 2016.

[26] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. In *Proc. Network and Distributed System Security Symposium (NDSS)*, page 14 pages., February 2016. doi: http://dx.doi.org/10.14722/ndss.2016.23187.

[27] Christian Decker and Roger Wattenhofer. Information Propagation in the Bitcoin Network. In *Proc. IEEE International Conference on Peer-to-Peer Computing*, pages 1–10, September 2013.

[28] Peter C. Dillinger and Panagiotis Manolios. Bloom filters in probabilistic verification. In *Proc. International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 367–381. Springer-Verlag, November 2004.

[29] Peter C. Dillinger and Panagiotis Manolios. Fast and Accurate Bitstate Verification for SPIN. In *Proc. Model Checking Software (SPIN)*, pages 57–75, April 2004. doi: 10.1007/978-3-540-24732-6_5.

[30] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM journal on computing*, 38(1):97–139, 2008.

[31] Thaddeus Dryja and Joseph Poon. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments. `https://lightning.network/lightning-network-paper.pdf`, Jan 2016.

[32] Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Olivier Mastropietro, Alex Lamb, Martin Arjovsky, and Aaron Courville. Adversarially learned inference. *arXiv preprint arXiv:1606.00704*, 2016.

[33] David Eppstein, Michael T. Goodrich, Frank Uyeda, and George Varghese. What's the Difference? Efficient Set Reconciliation Without Prior Context. In *Proc. ACM SIGCOMM*, pages 218–229, August 2011.

[34] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. Cuckoo filter: Practically better than bloom. In *Proc. ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT)*, pages 75–88, December 2014. doi: 10.1145/2674005.2674994. URL `http://doi.acm.org/10.1145/2674005.2674994`.

[35] Michael Fischer, Nancy Lynch, and Michael Paterson. Impossibility of distributed consensus with one faulty process. *JACM*, 32(2):374–382, 1985.

[36] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *Journal of Computer and System Sciences*, 31(2):182–209, 1985. ISSN 0022-0000. doi: https://doi.org/10.1016/0022-0000(85)90041-8. URL `http://www.sciencedirect.com/science/article/pii/0022000085900418`.

[37] Javier Garcia and Fernando Fernandez. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research*, 16(1):1437–1480, 2015.

[38] Mohammad Ghavamzadeh, Marek Petrik, and Yinlam Chow. Safe policy improvement by minimizing robust baseline regret. In *Advances in Neural Information Processing Systems*, pages 2298–2306, 2016.

[39] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proc. Symposium on Operating Systems Principles (SOSP)*, pages 51–68, October 2017.

[40] Shyamnath Gollakota and Dina Katabi. Zigzag decoding: Combating hidden terminals in wireless networks. In *Proc. ACM SIGCOMM Conference on Data Communication*, pages 159–170, August 2008. doi: 10.1145/1402958.1402977. URL `http://doi.acm.org/10.1145/1402958.1402977`.

[41] Solomon W. Golomb. Run-length encodings (Corresp.). *IEEE Transactions on Information Theory*, 12(3):399–401, July 1966. doi: 10.1109/TIT.1966.1053907.

[42] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680, 2014.

[43] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[44] Michael Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *Conf. on Comm., Control, and Computing*, pages 792–799, Sept 2011.

[45] Zhaohan Guo, Philip S. Thomas, and Emma Brunskill. Using options and covariance testing for long horizon off-policy policy evaluation. In *Advances in Neural Information Processing Systems*, pages 2492–2501, 2017.

[46] Anupam Gupta, Tomer Koren, and Kunal Talwar. Better algorithms for stochastic bandits with adversarial corruptions. *arXiv preprint arXiv:1902.08647*, 2019.

[47] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963.

[48] Wassily Hoeffding. *The collected works of Wassily Hoeffding*. Springer Science & Business Media, 2012.

[49] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies. *arXiv preprint arXiv:1702.02284*, 2017.

[50] Mark Huber. Halving the bounds for the markov, chebyshev, and chernoff inequalities using smoothing. *The American Mathematical Monthly*, 126(10): 915–927, 2019.

[51] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 652–661. PMLR, 2016.

[52] Kwang-Sung Jun, Lihong Li, Yuzhe Ma, and Jerry Zhu. Adversarial attacks on stochastic bandits. In *Advances in Neural Information Processing Systems*, pages 3640–3649, 2018.

[53] Sunny Katkuri. Improve block transfer efficiency using Graphene #17724. `https://github.com/ethereum/go-ethereum/pull/17724`, September 20 2018.

[54] Adam Kirsch and Michael Mitzenmacher. Less Hashing, Same Performance: Building a Better Bloom Filter. *Random Structures & Algorithms*, 33(2):187–218, September 2006. doi: 10.1002/rsa.20208.

[55] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *Proc. USENIX Security Symposium*, pages 279–296, August 2016.

[56] Eleftherios Kokoris-Kogiasy, Philipp Jovanovicy, Linus Gassery, Nicolas Gaillyy, Ewa Syta, and Bryan Ford. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In *Proc. IEEE Symposium on Security and Privacy*, pages 583–598, May 2018.

[57] Matthieu Komorowski, Leo A. Celi, Omar Badawi, Anthony C. Gordon, and A. Aldo Faisal. The artificial intelligence clinician learns optimal treatment strategies for sepsis in intensive care. *Nature Medicine*, 24(11):1716, 2018.

[58] Jernej Kos and Dawn Song. Delving into adversarial attacks on deep policies. *arXiv preprint arXiv:1705.06452*, 2017.

[59] Boris P. Kovatchev, Marc Breton, Chiara Dalla Man, and Claudio Cobelli. In silico preclinical trials: A proof of concept in closed-loop control of type 1 diabetes, 2009.

[60] Ilja Kuzborskij, Claire Vernade, András György, and Csaba Szepesvári. Confident off-policy evaluation and selection through self-normalized importance weighting. *arXiv preprint arXiv:2006.10460*, 2020.

[61] James Larisch, David Choffnes, Dave Levin, Bruce M Maggs, Alan Mislove, and Christo Wilson. CRLite: A Scalable System for Pushing All TLS Revocations to All Browsers. In *Proc. IEEE Symposium on Security and Privacy*, pages 539–556, May 2017.

[62] Romain Laroche, Paul Trichelair, and Remi Tachet Des Combes. Safe policy improvement with baseline bootstrapping. In *International Conference on Machine Learning*, pages 3652–3661, 2019.

[63] Michel Ledoux. *The concentration of measure phenomenon*. Number 89. American Mathematical Soc., 2001.

[64] Brian Levine and Gavin Andresen. IBLT Optimization. `https://github.com/umass-forensics/IBLT-optimization`, August 2018.

[65] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In *Proc. International Conference on Financial Cryptography and Data Security*, pages 528–547, Jan 2015.

[66] Shiau Hong Lim, Huan Xu, and Shie Mannor. Reinforcement learning in robust markov decision processes. In *Advances in Neural Information Processing Systems*, pages 701–709, 2013.

[67] Xihong Lin, Christian Genest, David L Banks, Geert Molenberghs, David W Scott, and Jane-Ling Wang. *Past, present, and future of statistical science*. CRC Press, 2014.

[68] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier, 1994.

[69] Fang Liu and Ness Shroff. Data poisoning attacks on stochastic bandits. *arXiv preprint arXiv:1905.06494*, 2019.

[70] Qiang Liu, Lihong Li, Ziyang Tang, and Dengyong Zhou. Breaking the curse of horizon: Infinite-horizon off-policy estimation. In *Advances in Neural Information Processing Systems*, pages 5356–5366, 2018.

[71] Eric Russell Love. Some Logarithm Inequalities. *The Mathematical Gazette (The Mathematical Association)*, 63(427):55–57, March 1980. URL `https://www.jstor.org/stable/3615890`.

[72] Lailong Luo, Deke Guo, Richard T.B. Ma, Ori Rottenstreich, and Xueshan Luo. Optimizing bloom filter: Challenges, solutions, and comparisons. *IEEE Communications Surveys Tutorials* (see also arXiv:1804.04777), 21(2):1912–1949, Second quarter 2019.

[73] A. Rupam Mahmood, Hado P. Van Hasselt, and Richard S Sutton. Weighted importance sampling for off-policy learning with linear function approximation. In *Advances in Neural Information Processing Systems*, pages 3014–3022, 2014.

[74] Chiara Dalla Man, Francesco Micheletto, Dayu Lv, Marc Breton, Boris Kovatchev, and Claudio Cobelli. The UVA/PADOVA type 1 diabetes simulator: New features. *Journal of Diabetes Science and Technology*, 8(1):26–34, 2014.

[75] Travis Mandel, Yun-En Liu, Sergey Levine, Emma Brunskill, and Zoran Popovic. Offline policy evaluation across representations with applications to educational games. In *AAMAS*, pages 1077–1084, 2014.

[76] Eduard Marin, Dave Singelée, Flavio D Garcia, Tom Chothia, Rik Willems, and Bart Preneel. On the (in) security of the latest generation implantable cardiac defibrillators and how to secure them. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 226–236. ACM, 2016.

[77] Andreas Maurer and Massimiliano Pontil. Empirical bernstein bounds and sample variance penalization. *arXiv preprint arXiv:0907.3740*, 2009.

[78] Ralph C. Merkle. A digital signature based on a conventional encryption function. In *Advances in Cryptology (CRYPTO '87)*, pages 369–378, August, 1987. doi: https://doi.org/10.1007/3-540-48184-2_32.

[79] Blossom Metevier, Stephen Giguere, Sarah Brockman, Ari Kobren, Yuriy Brun, Emma Brunskill, and Philip S Thomas. Offline contextual bandits with high probability fairness guarantees. In *Advances in Neural Information Processing Systems*, pages 14893–14904, 2019.

[80] Yaron Minsky, Ari Trachtenberg, and Richard Zippel. Set reconciliation with nearly optimal communication complexity. *IEEE Transactions on Information Theory*, 49(9):2213–2218, 2003.

[81] Michael Mitzenmacher and Rasmus Pagh. Simple multi-party set reconciliation. *Distributed Computing*, 31(6):441—453, November 2018.

[82] Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis*. Cambridge University Press, 2005. doi: 10.1017/CBO9780511813603.005.

[83] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[84] Michael Molloy. The Pure Literal Rule Threshold and Cores in Random Hypergraphs. In *Proc. ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 672–681, January 2004. URL `http://dl.acm.org/citation.cfm?id=982792.982896`.

[85] Jun Morimoto and Kenji Doya. Robust reinforcement learning. *Neural Computation*, 17(2):335–359, 2005.

[86] Rajeev Motwani and Prabhakar Raghavan. *Randomized algorithms*. Cambridge university press, 1995.

[87] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System, May 2009. URL `https://bitcoin.org/bitcoin.pdf`.

[88] Arnab Nilim and Laurent El Ghaoui. Robust control of markov decision processes with uncertain transition matrices. *Operations Research*, 53(5):780–798, 2005.

[89] A. Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, and Brian Neil Levine. Graphene: A New Protocol for Block Propagation Using Set Reconciliation. In *Proc. of International Workshop on Cryptocurrencies and Blockchain Technology (ESORICS Workshop)*, pages 420–428, September 2017.

[90] A Pinar Ozisik, Gavin Andresen, Brian N Levine, Darren Tapp, George Bissias, and Sunny Katkuri. Graphene: efficient interactive set reconciliation applied to blockchain propagation. In *Proceedings of the ACM Special Interest Group on Data Communication*, pages 303–317. 2019.

[91] Pinar Ozisik and Philip S Thomas. Security analysis of safe and seldonian reinforcement learning algorithms. *Advances in Neural Information Processing Systems*, 33, 2020.

[92] Jonathan Petit, Bas Stottelaar, Michael Feiri, and Frank Kargl. Remote attacks on automated vehicles sensors: Experiments on camera and lidar. *Black Hat Europe*, 11:2015, 2015.

[93] Lerrel Pinto, James Davidson, and Abhinav Gupta. Supervision via competition: Robot adversaries for learning tasks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1601–1608. IEEE, 2017.

[94] Lerrel Pinto, James Davidson, Rahul Sukthankar, and Abhinav Gupta. Robust adversarial reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2817–2826. JMLR. org, 2017.

[95] Salvatore Pontarelli, Pedro Reviriego, and Michael Mitzenmacher. Improving the performance of Invertible Bloom Lookup Tables. *Information Processing Letters*, 114(4):185–191, 2014. doi: https://doi.org/10.1016/j.ipl.2013.11.015.

[96] Doina Precup. Temporal abstraction in reinforcement learning. 2001.

[97] Maxim Raginsky and Igal Sason. Concentration of measure inequalities in information theory, communications and coding. *arXiv preprint arXiv:1212.4663*, 2012.

[98] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. Epopt: Learning robust neural network policies using model ensembles. *arXiv preprint arXiv:1610.01283*, 2016.

[99] Stephen B. Seidman. Network structure and minimum degree. *Social Networks*, 5(3):269–287, 1983. ISSN 0378-8733. doi: https://doi.org/10.1016/0378-8733(83)90028-X. URL `http://www.sciencedirect.com/science/article/pii/037887338390028X`.

[100] Rajneesh Sharma and Madan Gopal. A robust markov game controller for nonlinear systems. *Applied Soft Computing*, 7(3):818–827, 2007.

[101] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in Bitcoin. In *Proc. Financial Cryptography and Data Security*, pages 507–527, January 2015.

[102] Andrea Suisani, Andrew Clifford, Andrew Stone, Erik Beijnoff, Peter Rizun, Peter Tschipper, Alexandra Fedorova, Chen Feng, Victoria Lemieux, and Stefan Matthews. Measuring maximum sustained transaction throughput on a global network of Bitcoin nodes. In *Proc. Scaling Bitcoin Conference*, November 2017.

[103] Michel Talagrand. A new look at independence. *The Annals of probability*, pages 1–34, 1996.

[104] Sasu Tarkoma, Christian Esteve Rothenberg, and Eemil Lagerspetz. Theory and practice of bloom filters for distributed systems. *IEEE Communications Surveys Tutorials*, 14(1):131–155, First 2012. ISSN 1553-877X. doi: 10.1109/SURV.2011.031611.00024.

[105] P. S. Thomas, G. Theocharous, and M. Ghavamzadeh. High confidence policy improvement. In *International Conference on Machine Learning*, 2015.

[106] Philip S. Thomas. *Safe Reinforcement Learning*. PhD thesis, University of Massachusetts Libraries, 2015.

[107] Philip S. Thomas, Georgios Theocharous, and Mohammad Ghavamzadeh. High-confidence off-policy evaluation. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 2015.

[108] Philip S. Thomas, Bruno Castro da Silva, Andrew G. Barto, Stephen Giguere, Yuriy Brun, and Emma Brunskill. Preventing undesirable behavior of intelligent machines. *Science*, 366(6468):999–1004, 2019.

[109] Jonathan Toomim. Block propagation data from Bitcoin Cash's stress test. `https://medium.com/@j_73307/block-propagation-data-from-bitcoin-cashs-stress-test-5b1d7d39a234`, September 2018.

[110] Jonathan Toomim. Benefits of LTOR in block entropy encoding. https://medium.com/@j_73307/benefits-of-ltor-in-block-entropy-encoding-or-8d5b77cc2ab0, September 2018.

[111] Peter Tschipper. BUIP010 Xtreme Thinblocks. https://bitco.in/forum/threads/buip010-passed-xtreme-thinblocks.774/, Jan 2016.

[112] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust markov decision processes. *Mathematics of Operations Research*, 38(1):153–183, 2013.

[113] Gavin Wood. Ethereum: A secure decentralised generalised transaction ledger. https://ethereum.github.io/yellowpaper/paper.pdf, June 2018.

[114] Pieter Wuille. Minisketch: a library for bch-based set reconciliation. https://github.com/sipa/minisketch/blob/master/doc/math.md, 2018.

[115] Jinyu Xie. *Simglucose v0.2.1 (2018)*, 2019. URL https://github.com/jxx123/simglucose. Accessed May 1, 2020.

[116] Lin Yang, , Mohammad H. Hajiesmaili, M. Sadegh Talebi, John C. S. Lui, and Wing S. Wong. Adversarial bandits with corruptions: Regret lower bound and no-regret algorithm. In *Advances in Neural Information Processing Systems*, 2020.

[117] Julian Zimmert and Yevgeny Seldin. An optimal algorithm for stochastic and adversarial bandits. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 467–475. PMLR, 2019.