# LEARNING AND INCENTIVES IN COMPUTER SCIENCE

A DISSERTATION
SUBMITTED TO THE DEPARTMENT OF COMPUTER SCIENCE
AND THE COMMITTEE ON GRADUATE STUDIES
OF STANFORD UNIVERSITY
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Okke Schrijvers
June 2017

Okke Schrijvers

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Tim Roughgarden) Principal Adviser

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Dan Boneh)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Ashish Goel)

I certify that I have read this dissertation and that, in my opinion, it is fully adequate in scope and quality as a dissertation for the degree of Doctor of Philosophy.

(Nicolas Lambert)

Approved for the Stanford University Committee on Graduate Studies

# Abstract

In this thesis we look at topics in algorithmic game theory, with influences from learning theory. We use tools and insights from game theory to look at areas in computer science, such as machine learning and the bitcoin blockchain, that have been developed incognizant of incentives for selfish behavior. Here we 1) show that there are incentives to behave in a way that's harmful for the system, 2) give mechanisms where the incentives for the individual and group are aligned, and 3) measure how harmful (having to account for) selfish behavior is. In particular we study the problem of online prediction with expert advice, where we show that when experts are selfish and care about their reputation, the design of incentive compatible algorithms is tightly coupled to strictly proper scoring rules. We give algorithms that have good regret guarantees, and we prove that it is not possible to match regret guarantees for honest experts. For Bitcoin, we show that the way rewards are shared in mining pools can lead to miners strategically hiding work to improve their payout. This holds true even in the absence of outside options for miners, such as other pools or solo mining. We give a novel reward sharing function that does not have perverse incentives, and analyze its performance analytically and through simulations.

On the other hand we look at how data can be used in a variety of game theory applications. We ask the questions 1) how can we use data to replace standard informational assumptions in auction theory, 2) how much data do we need for good results in this area, 3) how can we use data to learn about the utilities of agents when we observe behavior, and finally (as misrecorded data may lead algorithms astray) 4) how can we find anomalies in a data set in an unsupervised manner? For the first two questions we look at position auction environments where we give the first computationally efficient algorithm for i.i.d. bidders with irregular value distributions, that achieves revenue arbitrarily close to optimal using polynomial samples. Due to the low sample complexity, our approach leads to a no-regret algorithm in the online setting. To address the third question, we give a computationally efficient algorithm that computes, given a correlated equilibrium (which may be a pure or mixed Nash equilibrium), the set of utilities that are consistent with it. Finally, we give an unsupervised anomaly detection algorithm that runs in a stream, and we show its performance through experiments on real and synthetic data.

# Acknowledgments

I am very thankful for the many people who have helped and inspired me during the last five years. First and foremost, Tim Roughgarden has been an incredible academic advisor and co-author on many of the papers that have led to the chapters in this thesis. He has given me the freedom to explore a diverse set of topics, while guiding me in asking the right questions for each of them. Tim has had a profound impact in how I conduct research, and how I communicate my work to others.

I'm also very fortunate to have had the opportunity to work with, and learn from, all of my amazing co-authors. In alphabetic order they are: Dan Boneh, Joe Bonneau, Sudipto Guha, Volodymyr Kuleshov, Mohammad Mahdian, Nina Mishra, Tim Roughgarden, Gourav Roy, and Sergei Vassilvitskii.[1] The diversity of their expertise and approach to research has truly expanded my experience.

Beyond the work I did at Stanford over the past five years, I've had the good fortune to do summer internships at Google, Amazon and Facebook. Not only were these great opportunities to broaden my horizon, the internships have led to publications at KDD and ICML, and a job as core data scientist at Facebook after graduation.

Finally, you cannot finish a PhD unless you start a PhD. Bert Heesakkers planted the seed to pursue a Master's degree and eventually PhD. My professors at Technische Universiteit Eindhoven have been extremely generous in the time they spent on coaching me through the application process and writing letters on my behalf. Finally, I'm grateful for my friends and family, who have borne with me through this entire process. I could not have done this without their support.

Okke Schrijvers

*San Francisco*
*May 31, 2017*

[1]This list includes co-authors from papers that I have not included in the thesis, but where the majority of the work was written during my Ph.D. [Roughgarden and Schrijvers, 2016a], and [Mahdian et al., 2015]. I'm also grateful to my co-authors of [Buchin et al., 2013], [Schrijvers and van Wijk, 2013], and [Schrijvers et al., 2013] which were all published during my Ph.D. but for which the majority of the work was done while I was a Master's student at Technische Universiteit Eindhoven.

# Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

A modern individual spends much of their time interacting with and through online computer platforms. We communicate with each other on platforms like Facebook and iMessage, search online using services like Google, ask and answer questions on websites like Stack Overflow and Quora, and order a ride on Uber or Lyft. What makes these systems complex, is that a user does not merely interact with the platform, but also with the others users of the platform: a social network or a ride-sharing service only provides value to the user through interactions with other users.

Consequently, the experience of users is dependent on both the design of the platform, as well as the interaction with other users. This raises an important issue: how can a platform ensure that the interaction with other users will be beneficial to all parties involved? A priori it is not certain that users have the same goals when using a platform: passengers on a ridesharing service seek to minimize the cost for a ride, while a driver on the same platform seeks to maximize their profit. Similarly, users on a Q&A website like Stack Overflow may be very interested to get detailed help with their questions, but may not want to exert effort to answer other people's questions. A user who has agency in their actions is called an *agent*, and a common way to describe the fact that an agent's personal goals may not align with the goals of other agents is *self-interest*. Informally we can then pose the question: *How can we design platforms that are robust to self-interested behavior of agents?*

To answer that question, we need some way to reason about how agents interact with each other. The field of *game theory* provides a model for this. Originally developed by von Neumann and Morgenstern [1944] and Nash [1951], agents are thought of as players in a game, where each agent has several actions she can undertake.[1] The joint profile of actions of the agents leads to a common outcome, and each participant derives some (potentially negative) utility from this outcome. For example, a game

[1] In 1994 Reinhard Selten, John Nash, and John Harsanyi received the Nobel Memorial Prize in Economics for their foundational work on the theory of non-cooperative games.

could be rock-paper-scissors where there are two players, each with three possible actions. An outcome is the decision by both players, where the player who wins has utility of 1, and the player that loses a utility of $-1$. Game theory isn't limited to literal games: for example one could also model morning traffic as a game. Each player decides which route to take in the morning, the joint outcome is which cars are on which roads, and the utilities for each player are inversely related to the time it takes them to reach their destination. As is common in the field, we refer to all such interactions as *games*. In these situations it is clear that agents seek to improve their own utility, e.g. their own travel time, rather than some collective goal, e.g. average travel time for all road users.

In many cases, an agent has some private information that influences the utility that she has for outcomes in a game, for example, a bidder in an auction may have a maximum willingness-to-pay for an item, and a buyer on an e-commerce platform may have an opinion on the service of a seller on that platform. Soliciting this information is subject to the same notion of self-interest as before. An agent may not simply supply the information truthfully, but instead report whatever information maximizes her utility. The study of how to design algorithms that operate on private information held by agents is known as *mechanism design*.[2] Mechanism design gives us insight in how we should design systems that are robust to self-interested behavior of agents.

This thesis deals with topics in game theory and mechanism design, but with a definite computer science perspective. In the late 90s, with the advent of the internet, computer scientists became interested in connections between computer science and game theory and started working on a field that is now known as *algorithmic game theory*. What sets this field apart from traditional game theory and mechanism design is that it takes computational efficiency as a hard design constraint: a theory or algorithm that cannot be run in reasonable time on a computer holds little value in practice. In this thesis, all results are computationally efficient, and computational efficiency is of particular importance in Chapters 2, 3 and 4. On the other hand, algorithmic game theory uses insights from game theory to analyze and improve computer platforms that have previously been incognizant of incentive problems. This is the primary motivation in Chapters 5 and 6.

What makes online platforms so interesting, and what guides the research questions in this thesis, is another aspect that online platforms provide: the collection of relevant *data*. Data can be useful because it provides insight in what agents want, and thus it gives a designer the opportunity to optimize for different objectives. At the same time, we should be aware that data may come from agents, and therefore we may

[2]In 2007, Leonid Hurwicz, Eric Maskin, and Roger Myerson received the Nobel Memorial Prize in Economics for their foundational work on Mechanism Design.

need to design the way in which we elicit information. This motivates looking at the questions: *How can we use data to learn about participants?* and *How can we elicit data truthfully?*

There are many ways in which data is used to get insights, from exploratory data analysis to deep learning approaches. In this thesis we aim to give provable guarantees for algorithms that rely on data, so we mostly rely on techniques from learning theory.

## 1.1 Research Goals

The questions of how to use data to understand agents, and how to design systems that are robust to self-interested behavior are broad enough to entertain the academic community for the decades to come. We next focus on the specific goals that this thesis aims to achieve.

### 1.1.1 Goal 1: Understanding Agents Through Data

In this thesis we seek to use data in such a way that we can prove formal guarantees of the algorithms that use data. While there is much value in applied machine learning and exploratory data science, we do not focus on these areas in the present work. Instead we seek to develop tools that can be used across a variety of application areas and be able to answer questions like: "what can we say about unobserved utilities of agents based on observations of behavior?" and "what is the smallest amount of data we need for our algorithm to do as well as an algorithm that knows with perfect precision which distribution the data came from?" This means that we mostly use tools and insights from learning theory.

When agents have private information, it is fairly common in economics to assume that this private information comes from some underlying distribution which is known to all participants. One can then design mechanisms that take this distribution as an input or do equilibrium analysis based on the assumption that all participants know the distribution that other participants' types are drawn from. One reason why this may be a reasonable assumption is that this distribution may be approximated by an empirical distribution of past observations. However, it's not clear that guarantees that work well for the true distribution carry over without loss (or with minimal loss) to empirical distributions. In fact, in general this is not the case, as we give example of in Chapter 2. This motivates explicitly modeling the information about the underlying distribution in terms of samples (data) from that distribution:

**Question 1.** *For algorithms that rely on knowledge of a distribution $F$, can we replace knowledge of $F$ with access to samples from $F$?*

In the limit we expect the answer to the question to be affirmative: indeed if we have an infinite number of samples from a distribution, the empirical cumulative distribution converges to the actual cumulative distribution almost surely. Therefore, it is informative to not just ask whether it's possible to replace knowledge of a distribution with access to samples, but also how many samples are needed for good results:

**Question 2.** *How many samples from $F$ are necessary and sufficient to do (almost) as well as knowing $F$ directly (with high probability)?*

Answers to this question can give us insight into how useful this technique may be in practice (if an exponential number of samples are necessary, we could only use the approach in small settings), but it can also guide us to determine how much data is needed, and at which stage more data adds little in terms of performance guarantees.

Another question lies more fundamentally at the assumptions of where we get our input from to begin with. Taking an online platform as an example, we can typically directly observe what a user does, and how they are interacting with the platform and other users, but we cannot directly observe why they chose these actions, and what their utility from these actions was.

**Question 3.** *Can we learn the utilities of players in a game if we can only observe behavior?*

In both of the questions above, more data is generally better. However, in practical applications, with more data, we also run the risk that some of the data may be incorrect. For example, agents may have acted irrational at some point in time, e.g. to protect their privacy, or there may have been data corruption while recording data. The need to remove anomalous data exists regardless of the source of data, so we phrase the problem generally, without limiting ourselves to applications where incentives play an important role.

**Question 4.** *Given a data set, can we determine in an unsupervised manner, which (if any) elements may be anomalies?*

It may be difficult or impossible to prove theoretical guarantees on the effectiveness of anomaly detection without making strong assumptions on the data generating process. The algorithms that are used in practice lack these guarantees, but still seem to give good results. We therefore seek to evaluate the performance of an anomaly detection algorithm based on performance on real data, while at the same time we prove structural properties of the algorithm and data structure to give insight into why it may perform well.

Note that all of the questions above could also be phrased in terms of robustness. Algorithms whose guarantees break down when we only have access to an empirical input distribution are not very robust, but ones that provide guarantees with

high probability based on samples are. Being able to test if agents are playing an equilibrium, even if we don't know their utilities, can validate or reject rationality hypotheses. This is robustness to modeling assumptions. Finally, being able to identify, in an unsupervised way, which inputs are likely to be anomalies, can help us with algorithms that are more robust to outliers.

### 1.1.2 Goal 2: Robustness to Incentives

A different sense of robustness, is robustness with respect to the self-interested behavior of agents. In this part of the thesis we ask the question: which applications in computer science present incentives for the participants, but lack an analysis of how these incentives influence their behavior? As stated before, it is not obvious that a user will always act in accord with what's best for the platform as a whole, but rather is motivated by the value they get out of a system.

The common approach to algorithm design is to assume that the input is given, and that the goal of the algorithm designer is simply to compute an output that satisfies the purpose of the algorithm, e.g. an algorithm to find the highest value in a list, scans through the list and returns the highest value it has seen. However, if the input comes from agents, this approach may break down. Imagine we're trying to give free concert tickets to whoever would value them the most. If we were to ask a group of people to report how much they would value the tickets, everyone *could* truthfully report their true value for them (assuming they know this). However, if an agent wants these tickets, they might be inclined to name a higher number than their true value, to improve their chances of naming the highest number and thus getting the tickets. This motivates the following question:

**Question 5.** *For algorithms that operate on data given by agents, do there exist incentives to misreport the data?*

In the example given above, the answer is clearly affirmative. In fact, since people are unconstrained in their reports, this algorithm will result in a game of 'name the highest number,' with the recipient of the tickets unrelated to whoever would value them the most. Identifying algorithms that are subject to being gamed is useful in its own right, but ideally we'd still want to run some (other) algorithm that calculates the desired outcome:

**Question 6.** *For algorithms where incentives exist for data sources, are there alternative algorithms where agents are incentivized to report honestly?*

In the case of giving away concert tickets, one could for example ask participants to list their value in dollars, give the tickets to the person who names the highest dollar

amount, and charge them the second highest amount.[3] It seems like this solution is even better than what we started out with: not only do we give the tickets to the person who values them most, but we might make some money in the process. There are indeed many application areas where auctions are very successful, but there are also areas where there are moral or ethical objections to this. Consider for example the question of who should receive a donated kidney, or how medical residents should be assigned to hospitals. We don't want this to always go to the highest bidder. This motivates the question:

**Question 7.** *Are algorithms that incentivize honest reporting by agents as powerful as algorithms that have direct access to the data? I.e. is the presence of agents detrimental to the performance of a system?*

In the case of assigning an item to the person who values it the most, one can show that it's necessary to include a payment of some sort, in order to incentivize truthful reporting. In other cases we may have to resort to optimizing some objective function approximately rather than exactly. The performance metric in this question is going to differ depending on the application, but getting insight into the cost of designing systems to account for self-interested behavior can help in guiding what a system lets its users do.

We have motivated Questions 5-7 with an example from auction theory. But there are potentially many applications in computer science where the incentive question has simply not been considered yet. In this thesis we look at two such applications and provide answers to the three questions for both of them.

## 1.2 Contributions of the Thesis

This thesis is divided in two parts. In the first part we address the goal of understanding agents through data by looking at Questions 1-4. The second part of the thesis is about robustness to incentives, and we answer Questions 5-7 for two different applications. The chapters in this thesis can be read independently of each other.

### 1.2.1 Part I: Learning

Ad auctions are one of the predominant generators of revenue for internet companies like Google and Facebook. The economic theory of revenue-optimal auctions is has

[3]This is known in the auction literature as the second-price auction. Intuitively no person has a reason to misreport because they have to pay for taking an item that could have gone to someone else, but they do not directly influence the amount they pay. For a more detailed treatment, see e.g. [Roughgarden, 2016].

been well understood since the 80s, but it typically relies on strong assumptions, such as access to the precise distribution $F$ of 'willingness-to-pay' of bidders. In chapter 2, we relax this assumption, and propose an auction format that uses $n$ samples from $F$ (which we think of as past auction data) to get a $(1-\epsilon)$ fraction of the revenue with high probability, for i.i.d. bidders in matroid and position auctions. We give sample complexity bounds that show the relationship between the amount of data $n$ and the approximation factor $\epsilon$. Chapter 2 addresses Questions 1 and 2.

Auctions are an application area of game theory where the seller and bidders have very natural objectives. In other areas, this is less clear. Users of ride-share services may have different willingness-to-pay when demand for rides exceeds the supply of drivers, and a fanpage on social media may time their post to engagement in terms of likes, shares, comments or exposure. In both of these situations, the preferences of people are not known a priori, but insight in their preferences may improve the economic efficiency of the systems. In Chapter 3 we address Question 3 and describe a method to uncover the utility structure of players in games, based on observed play. On a technical level we show that succinct games, where players are playing a correlated equilibrium, can be rationalized in polynomial time. We also show that when the structure of the played game is not known, the problem becomes NP-hard.

These methods, for optimal auctions and rationalizing games, only have provable guarantees when the data comes from processes that agree with the assumptions that are made, and they tend to be more accurate with more available data. However, in the real world, as the amount of data grows, the probability that anomalous data enters our data set also increases. So in Chapter 4 we address Question 4 and give an unsupervised approach for outlier detection that works well for both high and low-dimensional data. We give streaming algorithms for maintaining the relevant data structure and doing anomaly detection, and our method compares favorably to existing methods.

### 1.2.2 Part II: Incentives

In other cases, we do not need data to understand what drives human behavior, as it can be directly derived from the kind of incentives that a computer system provides. One such example is in machine learning: prediction with expert advice is a classical problem in this area where the credibility or influence of an expert depends on her past predictions. In the real world, experts care about the influence they wield and so they may report untruthfully when so incentivized. Look no further than political pundits who are rewarded for extreme rather than nuanced claims, to see this behavior in action. False reports can impede an aggregator's ability to learn from experts. In Chapter 5 we address Questions 5-7 for the case of learning from expert advice: we quantify how harmful selfish behavior of experts can be, and give an algorithm

where experts are always incentivized to report truthfully. We also prove a separation between the best possible regret when experts are selfish and when they are honest (i.e. do not respond to incentives and just reveal private information) pointing at the harm of selfish behavior.

Another system where participants actions are driven by incentives, this time quite explicitly, is Bitcoin mining. To have a decentralized currency, Bitcoin processes transactions by a consensus protocol that relies on solving a cryptographic puzzle. This consumes computational resources, and so participants are compensated by monetary rewards. We show in Chapter 6 that even in very simple settings—only a single mining pool, without outside options—there are incentives to delay reporting of solutions to the puzzle, answering Question 5 for Bitcoin mining. Even in the case of a single pool this means computational resources are wasted, but when the pool is a part of a larger system the same incentives still apply and now all people in the pool run the risk of losing out on payments. We propose a new reward function that resolves this incentive problem (addressing Question 6), and show that it has nice properties for deployment in practice (Question 7).

At the end of the day, building a robust computer system is a complicated task. From rewarding desired behavior and disincentivizing unwanted behavior, to utilizing usage data to understand and improve the way your system works, a modern computer scientist needs a large toolset to tackle these problems. This thesis addresses some of these issues, and as more and more interactions move to the digital space, we are unlikely to run out of new problems to tackle soon.

# Part I

# Learning

# Chapter 2

# Learning Optimal Auctions

This chapter presents the first polynomial-time algorithm for position and matroid auction environments that learns, from samples from an unknown bounded valuation distribution, an auction with expected revenue arbitrarily close to the maximum possible.[1] In contrast to most previous work, our results apply to arbitrary (not necessarily regular) distributions and the strongest possible benchmark, the Myerson-optimal auction. Learning a near-optimal auction for an irregular distribution is technically challenging because it requires learning the appropriate "ironed intervals," a delicate global property of the distribution.

## 2.1 Introduction

The traditional economic approach to revenue-maximizing auction design, exemplified by Myerson [1981], posits a known prior distribution over what bidders are willing to pay, and then solves for the auction that maximizes the seller's expected revenue with respect to this distribution. Recently, there has been an explosion of work in computer science that strives to make the classical theory more "data-driven," replacing the assumption of a known prior distribution with that of access to relevant data, in the form of samples from an unknown distribution. In this paper, we study the problem of learning a near-optimal auction from samples, adopting the formalism of Cole and Roughgarden [2014]. The idea of the model, inspired by PAC-learning [Valiant, 1984], is to parameterize through samples the "amount of knowledge" that the seller has about bidders' valuation distributions.

We consider single-parameter settings, where each of $n$ bidders has a private valuation (i.e., willingness to pay) for "winning" and valuation 0 for "losing." Feasible outcomes correspond to subsets of bidders that can simultaneously win; the feasible

[1] The chapter is based on work presented at EC'16 [Roughgarden and Schrijvers, 2016b]

subsets are known in advance.[2] We assume that bidders' valuations are drawn i.i.d. from a distribution $F$ that is unknown to the seller. However, we assume that the seller has access to $m$ i.i.d. samples from the distribution $F$ — for example, bids that were observed in comparable auctions in the past. The goal is to design a polynomial-time algorithm $A(v_1, \ldots, v_m)$, mapping samples $v_i \sim F$ to truthful auctions, such that, for every distribution $F$, the expected revenue is at least $1-\epsilon$ times the optimal expected revenue.[3] The sample complexity of achieving a given approximation factor $1-\epsilon$ is then the minimum number of samples $m$ such that there exists a learning algorithm $A$ with the desired approximation. This model serves as a potential "sweet spot" between worst-case and average-case analysis, inheriting much of the robustness of the worst-case model (since we demand guarantees for every underlying distribution $F$) while allowing very good approximation guarantees.

### 2.1.1 Our Results

We give polynomial-time algorithms for learning a $(1-\epsilon)$-approximate auction from samples, for arbitrary matroid (or position) auction environments and valuation distributions that satisfy minimal assumptions.

For example, for valuation distributions with support in $[0, H]$, we provide a polynomial-time algorithm that, given a matroid environment with $n$ bidders and $m$ i.i.d. samples from an arbitrary distribution $F$, with probability $1-\delta$, approximates the maximum-possible expected revenue up to an additive loss of at most $3n\sqrt{\frac{\ln 2\delta^{-1}}{2m}} \cdot H$. Thus for every $\epsilon > 0$, the additive loss is at most $\epsilon$ (with probability at least $1-\delta$) provided $m = \Omega(n^2 H^2 \epsilon^{-2} \log \delta^{-1})$. Whenever the optimal expected revenue is bounded away from 0, this result immediately implies a comparable sample complexity bound for learning a $(1-\epsilon)$-(multiplicative) approximate auction. Our main result can also be used to give a no-regret guarantee in a stochastic learning setting (Section 2.5).

A lower bound of Cesa-Bianchi et al. [2015] implies that, already for simpler settings, the quadratic dependence of our sample complexity bound on $\frac{1}{\epsilon}$ is optimal. A lower bound of Huang et al. [2015] implies that, already with a single bidder, the sample complexity must depend polynomially on $H$. Whether or not the sample

[2]For example, in auction with $k$ copies of an item, where each bidder only wants one copy, feasible outcomes correspond to subsets of at most $k$ bidders.

[3]By a truthful auction, we mean one in which truthful bidding is a dominant strategy for every bidder. The restriction to dominant strategies is natural given our assumption of an unknown distribution. Given this, the restriction to truthful auctions is without loss of generality (by the "Revelation Principle," see e.g. [Nisan, 2007]). Also, for the single-parameter problems that we study, there is always an optimal auction in which all bidders have dominant strategies [Myerson, 1981].

complexity needs to depend on $n$ is an interesting open question.

Our technical approach is based on a "switching trick" (Proposition 2.3) and we believe it will lead to further applications. A key idea is to express the difference in expected revenue between an optimal and a learned auction in a matroid or position environment purely in terms of a difference in area under the true and estimated revenue curves. This "global" analysis avoids having to compare explicitly the true and estimated virtual valuations or the optimal and learned allocation rules. With this approach, there is clear motivation behind each of the steps of the learning algorithm, and the error analysis, while non-trivial, remains tractable in settings more general than those studied in most previous works.

The assumption of bounded valuations is not necessary for our results to hold. More generally, the only assumption required is that the optimal auction does not obtain a constant fraction of its expected revenue from valuation profiles with at least one extremely high-valued bidder (with valuation bigger than a parameter $H$). This assumption is trivially satisfied by any distribution with support in $[0, H]$, and is also satisfied (for a suitable $H$) by many (irregular) distributions with infinite support. Some assumption of this type is necessary to preclude pathological distributions that are impossible for any algorithm to learn.[4]

### 2.1.2 Why Irregular Distributions Are Interesting

A majority of the literature on approximation guarantees for revenue maximization (via learning algorithms or otherwise) restricts attention to "regular" valuation distributions or subclasses thereof; see related work below for examples and exceptions. Formally, a distribution $F$ with density $f$ is *regular* if

$$\varphi(v) = v - \frac{1 - F(v)}{f(v)} \tag{2.1}$$

is a nondecreasing function of $v$. $\varphi$ is also called the *virtual valuation* function. Intuitively, regularity is a logconcavity-type assumption that provides control over the tail of the distribution. While many important distributions are regular, plenty of natural distributions are not. For example, Sivan and Syrgkanis [2013] point out that mixtures of distributions (even of uniform distributions) tend to be irregular, and yet are obviously prevalent in the real world.

[4]To appreciate this issue, consider a single-bidder problem and all distributions that take on a value $M^2$ with probability $\frac{1}{M}$ and 0 with probability $1 - \frac{1}{M}$. The optimal auction for such a distribution earns expected revenue at least $M$. It is not difficult to prove that, for every $m$, there is no way to use $m$ samples to achieve near-optimal revenue for every such distribution — for sufficiently large $M$, all $m$ samples are 0 w.h.p. and the algorithm has to resort to an uneducated guess for $M$.

### 2.1.3 Why Irregular Distributions Are Hard

To understand why irregular distributions are so much more technically challenging than regular distributions, we need to review some classical optimal auction theory. We can illustrate the important points already in single-item auctions. Myerson [1981] proved that, for every regular distribution $F$, the optimal auction is simply a second-price auction supplemented with a reserve price of $\varphi^{-1}(0)$, where $\varphi$ denotes the virtual valuation function in (2.1). (The winner, if any, pays the higher of the reserve price and the second-highest bid.) Thus, *learning the optimal auction reduces to learning the optimal reserve price*, a single statistic of the unknown distribution. And indeed, for an unknown regular distribution $F$, there is a polynomial-time learning algorithm that needs only $\text{poly}(\frac{1}{\epsilon})$ samples to compute a $(1-\epsilon)$-approximate auction [Dhangwatnotai et al., 2010, Huang et al., 2015].

The technical challenge of irregular distributions is the need to *iron*. When the virtual valuation function $\varphi$ of the distribution $F$ is not nondecreasing, Myerson [1981] gave a recipe for transforming $\varphi$ into a nondecreasing "ironed" virtual valuation function $\overline{\varphi}$ such that the optimal single-item auction awards the item to the bidder with the highest positive ironed virtual valuation (if any), breaking ties randomly (or lexicographically). Intuitively, this ironing procedure identifies intervals of non-monotonicity in $\varphi$ and changes the value of the function to be constant on each of these intervals. (See also below and the exposition by Hartline [2014].)

The point is that the appropriate ironing intervals of a distribution are a *global* property of the distribution and its (unironed) virtual valuation function. Estimating the virtual valuation function at a single point — all that is needed in the regular case — would appear much easier than estimating the right intervals to iron in the irregular case.

We present two examples to drive this point home. The first, which is standard, shows that foregoing all ironing can lead to a constant-factor loss in expected revenue, even in single-item auctions. (Reserve prices are still worse in matroid environments, see Devanur et al. [2014].) The second example shows that tiny mistakes in the choice of ironing intervals can lead to a large loss of expected revenue.

**Example 2.1** (Ironing Is Necessary for Near-Optimal Revenue)**.** *The distribution is as follows: with probability* $1/H$ *the value is* $H$ *(for a large* $H$*) and it is* $1$ *otherwise. The optimal auction irons the interval* $[1, H)$ *for expected revenue of* $2 - \frac{1}{n}$ *[Hartline, 2014], which approaches 2 with many bidders* $n$*. Auctions that do not implicitly or explicitly iron obtain expected revenue only 1.*

**Example 2.2** (Small Mistakes Matter)**.** *Let* $F$ *be* $5$ *with probability* $1/10$ *and* $1$ *otherwise, and consider a single-item auction with* $10$ *bidders. The optimal auction irons the interval* $[1, 5)$ *and has no reserve price. If there are at least two bidders with value* $5$ *one of them will get the item at price* $5$*; if all bidders have value* $1$*, one of them*

*will receive it at price* 1*. If there is exactly one bidder with value* 5*, then her price is* $\frac{1}{10} \cdot 1 + \frac{9}{10} \cdot 5 = \frac{46}{10}$.

*Now consider an algorithm that slightly* overestimated *the end of the ironing interval to be* $[1, 5 + \epsilon)$ *with* $\epsilon > 0$*. (Imagine* $F$ *actually has small but non-zero density above 5, so that this mistake could conceivably occur.) Now all bids always fall in the ironing interval and therefore the item is always awarded to one of the players at price* 1*. Not only do we lose revenue when there is exactly one high bidder, but additionally we lose revenue for auctions with at least two bidders with value* 5*. This auction has even worse revenue than a Vickrey (second-price) auction, so the attempt to iron did more harm than good.*

Now consider the same setting as Example 2.2, with the exception that we slightly underestimate the ironing interval as $[1, 5-\epsilon)$, instead of overestimating. We still lose some revenue compared to the optimal ironing interval —namely when there is one high bidder, she pays $\frac{46}{50} - \frac{9}{10}\epsilon$ instead of $\frac{46}{50}$— but the revenue is much closer to optimal than when the end point of the ironing interval was too high. This phenomenon, that underestimation is better than overestimation, is true more generally. Our learning algorithm deliberately reports ironing intervals (and a reserve price) that are slightly lower than the data would suggest, to guarantee that with high probability the start and end points of ironing intervals do not exceed the optimal such points.

### 2.1.4 Related Work

Elkind [2007] gives a polynomial-time learning algorithm for the restricted case of single-item auctions with discrete distributions with known finite supports but with unknown probabilities. Learning is done using an oracle that compares the expected revenue of pairs of auctions, and $O(n^2K^2)$ oracle calls suffice to determine the optimal auction (where $n$ is the number of bidders and $K$ is the support size of the distributions). Elkind [2007] notes that such oracle calls can be implemented approximately by sampling (with high probability), but no specific sample complexity bounds are stated.

Cole and Roughgarden [2014] also give a polynomial-time algorithm for learning a $(1 - \epsilon)$-approximate auction for single-item auctions with non-identical bidders, under incomparable assumptions to [Elkind, 2007]: valuation distributions that can be unbounded but must be regular. It is necessary and sufficient to have $m = \text{poly}(n, \frac{1}{\epsilon})$ samples, however in the analysis by Cole and Roughgarden [2014] the exponent in the upper bound is large (currently, 10). These sample complexity results were recently generalized and improved dramatically by Devanur et al. [2016], although though at the time of publication, all of their results still require the valuation distributions to be regular.

Meanwhile, Morgenstern and Roughgarden [2015] gave general sample complexity upper bounds which are similar to ours and cover all single-parameter settings (matroid and otherwise), although their (brute-force) learning algorithms are not computationally efficient.

After this present work was published, Gonczarowski and Nisan [2017] used an $\epsilon$-net in value space to give a computationally efficient algorithm with polynomial sample complexity for most single-parameter environments (including non-i.i.d., non-regular environments), and Devanur et al. [2016] extended their own work to handle some non-regular settings as well.

The papers of Cesa-Bianchi et al. [2015] and Medina and Mohri [2014] give algorithms for learning the optimal reserve-price-based single-item auction. Recall from Example 2.1 that, with irregular distributions, the expected revenue of the best reserve-price-based auction might be only half that of an optimal auction.

Dughmi et al. [2014] proved negative results (exponential sample complexity) for learning near-optimal mechanisms in multi-parameter settings that are much more complex than the single-parameter settings studied here. The paper also contains positive results for restricted classes of mechanisms.

Huang et al. [2015] give optimal sample complexity bounds for the special case of a single bidder under several different distributional assumptions, including for the case of bounded irregular distributions where they need $O(H \cdot \epsilon^{-2} \cdot \log(H\epsilon^{-1}))$ samples.

Our learning algorithm is in the spirit of the Random Sampling Empirical Myerson mechanism [Devanur et al., 2014] and its precursors, though different in a number of details. Previous work used the approach to construct a revenue curve from bidders in an auction and prove constant-factor approximations in prior-free settings. The present work seeks $(1 - \epsilon)$-approximations in settings with an unknown distribution.

For previously studied models about revenue-maximization with an unknown distribution, which differ in various respects from the model of Cole and Roughgarden [2014], see Babaioff et al. [2011], Cesa-Bianchi et al. [2015], and Kleinberg and Leighton [2003]. For other ways to parameterize partial knowledge about valuations, see e.g. [Azar et al., 2013] and [Chiesa et al., 2012]. For other ways to parameterize a distribution by its "degree of irregularity" see [Hartline, 2014], [Huang et al., 2015], and [Sivan and Syrgkanis, 2013]. For other uses of samples in auction design that differ from ours, see [Fu et al., 2014], who use samples to extend the Crémer-McLean theorem [Crémer and McLean, 1985] to partially known valuation distributions, and [Chawla et al., 2014], which is discussed further below. For asymptotic optimality results in various symmetric settings (single-item auctions, digital goods), which identify conditions under which the expected revenue of some auction of interest (e.g., second-price) approaches the optimal with an increasing number of i.i.d. bidders, see [Neeman, 2003], [Segal, 2003], [Baliga and Vohra, 2003], and [Goldberg et al., 2006].

For applications of learning theory concepts to prior-free auction design in unlimited-supply settings, see [Balcan et al., 2008].

Finally, the technical issue of ironing from samples comes up also in [Ha and Hartline, 2013] and [Chawla et al., 2014], in models incomparable to the one studied here. The setting of [Ha and Hartline, 2013] is constant-factor approximation guarantees for prior-free revenue maximization, where the goal is input-by-input rather than distribution-by-distribution guarantees. Chawla et al. [2014] study non-truthful auctions, where bidders' true valuations need to be inferred from equilibrium bids, and aim to learn the optimal "rank-based auction," which can have expected revenue a constant factor less than that of an optimal auction. Our goal of obtaining a $(1-\epsilon)$-approximation of the maximum revenue achieved by any auction is impossible in both of these settings.

Summarizing, this chapter details the first polynomial-time algorithm for position and matroid environments that learns, from samples from an unknown irregular valuation distribution, an auction with expected revenue arbitrarily close to the maximum possible.

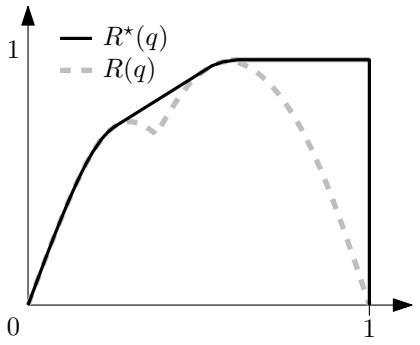
## 2.2 Preliminaries

### 2.2.1 The Empirical CDF and the DKW Inequality

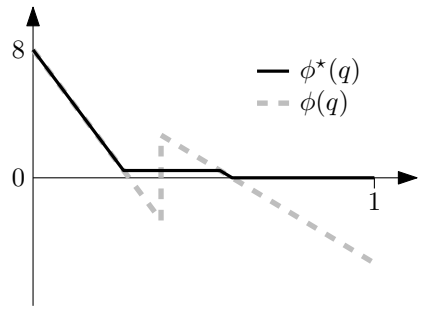
Let $X = \{X_i\}_{i=1}^m$ be a set of $m$ samples, and let $X^{(i)}$ be the $i^{\text{th}}$ order statistic. We use the standard notion of the empirical cumulative distribution function (empirical CDF): $\widehat{F_m}(v) = \frac{1}{m} \cdot |\{X_i : X_i \leq v\}|$. The empirical CDF is an estimator for the quantile of a given value. The Dvoretzky-Kiefer-Wolfowitz (DKW) inequality [Dvoretzky et al., 1956, Massart, 1990] states that the difference between the empirical CDF and the actual CDF decreases quickly in the number of samples. Let $\epsilon_{m,\delta} = \sqrt{\frac{\ln 2\delta^{-1}}{2m}}$, then $\Pr\left[\sup_{v\in\mathbb{R}} \left|F(v) - \widehat{F_m}(v)\right| \leq \epsilon_{m,\delta}\right] \geq 1-\delta$. So the largest error in the empirical CDF shrinks in $O(m^{-1/2})$. For our purposes we will not need the CDF $F$, but rather its inverse $F^{-1}$. Define $\widehat{F_m^{-1}}(x)$ as $X^{(\max(1,\lceil x\cdot m\rceil))}$ for $x \in [0,1]$. (For convenience, define $\widehat{F_m^{-1}}(x)$ as 0 if $x < 0$ and $H$ if $x > 1$.) By definition, for all $v \in [0, H]$:

$$\widehat{F_m^{-1}}\left(\widehat{F_m}(v)\right) \quad \leq \quad v \quad \leq \quad \widehat{F_m^{-1}}\left(\widehat{F_m}(v) + \frac{1}{m}\right). \tag{2.2}$$

In the remainder of this paper, we will use $\widehat{F}$, $\widehat{F^{-1}}$, and $\epsilon$ without explicitly referring to the number of samples $m$ and confidence parameter $\delta$.




(a) Revenue curve $R(q)$ and the optimal induced revenue curve $R^\star(q)$.




(b) Virtual value function $\varphi(q)$ and the induced virtual value function $\varphi^\star(q)$.

Figure 2.1: The dashed gray line is the revenue curve $R(q)$ and its derivative, the virtual value function $\varphi(q)$, for a irregular bimodal distribution. In black we have the curves $\varphi^{opt}$ and $R^{opt}$ corresponding to the optimal ironing interval, and optimal reserve price for this distribution. This example taken from [Hartline, 2014, Chapter 3].

### 2.2.2 Optimal Auctions using the Revenue Curve

**Revenue Curve and Quantile Space** For a value $v$ and a distribution $F$, we define the corresponding *quantile* as $1 - F(v)$.[5] The quantile of $v$ corresponds to the sale probability for a single bidder with valuation drawn from $F$ who faces a posted price $v$. Note that higher values $v$ correspond to lower quantiles. The *revenue curve* $R$ of a valuation distribution $F$ is the curve, in quantile space, defined by $R(q) = q \cdot F^{-1}(1-q)$ (Figure 2.1a). Myerson [1981] showed that the ex ante expected revenue for a bidder in a truthful auction is $\mathbb{E}_{q\sim U[0,1]}[-R(q) \cdot y'(q)]$, where $y$ is the interim allocation function (defined formally below) for that auction. The derivative of $R$ is the virtual value function $\varphi(.)$ — recall (2.1) — in quantile space, see Figure 2.1b. For regular distributions the revenue curve is concave and the virtual value function non-increasing, but for irregular distributions this is not the case.

Myerson [1981] showed that for any non-concave revenue curve $R$, one can create an allocation rule that will yield the same revenue as $R$'s convex hull $\mathcal{CH}(R)$. This procedure is called *ironing*, and for each interval where $R$ differs from $\mathcal{CH}(R)$, we define the *ironed virtual value* to be slope of the convex hull over this interval. This means the virtual values are equal in this interval, and hence any two bids in that range are interchangeable, and so an iron-based allocation function is constant on this interval.[6]

[5] This is for consistency with recent literature; "reversed quantile" might be a more accurate term.

[6] It takes some effort to show that keeping the allocation probability constant on an interval has exactly the effect we described here [Myerson, 1981].

The resulting *ironed* revenue curve $R^\star$ will be concave and the corresponding ironed virtual value function $\varphi^\star$ will be non-decreasing. It is also useful to think of a reserve price $r$ (with corresponding quantile $q_r$) in a similar way, as effectively changing the virtual valuation function so that $\varphi^\star(q) = 0$ whenever $q \geq q_r$ (Figure 2.1b), with the corresponding revenue curve $R^\star$ constant in that region (Figure 2.1a).[7]

More generally, given any set of disjoint ironing intervals $\mathcal{I}$ and reserve price $r$, both in value space, we can imagine the effect on the revenue curve as follows. (For now this is a thought experiment; Proposition 2.3 connects these revenue curve modifications back to allocation rule modifications.) Let $R$ be the revenue curve without ironing or a reserve price, and define $R^{(\mathcal{I},r)}$ as the revenue curve *induced* by a set $\mathcal{I}$ of ironing intervals and reserve price $r$. This curve is defined by

$$R^{(\mathcal{I},r)}(q) = \begin{cases} R(F(r)) & \text{if } q > F(r) \\ R(q_a) + \frac{q-q_a}{q_b-q_a}\left(R(q_b) - R(q_a)\right) & \text{if } q \in (q_a, q_b] \\ & \quad \text{with } [F^{-1}(1-q_b), F^{-1}(1-q_a)) \in \mathcal{I} \\ R(q) & \text{otherwise.} \end{cases} \tag{2.3}$$

Given $\mathcal{I}$ and $r$ as above, we define the auction $A^{(\mathcal{I},r)}$ as follows: given a bid profile: (i) reject every bid with $b_i < r$; (ii) for each ironing interval $[a, b) \in \mathcal{I}$, treat all bids $\{b_j : a \leq b_j < b\}$ as identical (equal to some common number between $a$ and $b$); (iii) among the remaining bidders, maximize the sum of the ironed bids of the winners; (iv) charge payments so that losers always pay 0 and so that truthful bidding is a dominant strategy for every player. This auction is well defined (i.e., independent of the choice of the common numbers in (ii)) in settings where the computation in (iii) depends only on the ordering of the ironed bids, and not on their numerical values. In this case, the payments in (iv) are uniquely defined (by standard mechanism design results). This is the case in every matroid environment[8] and also in position auctions.[9] In such a setting, we use $\mathcal{A}$ to denote the set of all auctions of the form $A^{(\mathcal{I},r)}$. We restrict attention to such settings in the remainder of the paper.

[7]Most of the existing literature would not consider the effect of the reserve price on the revenue curve, in which case the black and dashed lines would coincide after the second peak. However, by including its effect as we did, we'll be able to apply the Switching Trick described below.

[8]In a matroid environment, the set $\mathcal{F}$ of feasible outcomes satisfies: (i) (downward-closed) $T \in \mathcal{F}$ and $S \subseteq T$ implies $S \in F$; and (ii) (exchange property) whenever $S, T \in \mathcal{I}$ with $|T| < |S|$, there is some $i \in S \setminus T$ such that $T \cup \{i\} \in \mathcal{I}$.

[9]In a position auction, $n$ bidders vie for $k$ "slots," with at most one bidder assigned to each slot and at most one slot assigned to each bidder. Being assigned slot $j$ corresponds to an allocation amount $\alpha_j$, which historically corresponds to a "click-through rate." See [Edelman et al., 2007] and [Varian, 2007] for details.

**The Switching Trick** Given a distribution $F$, we explained two ways to use ironing intervals $\mathcal{I}$ and a reserve price $r$: (i) to define a modified revenue curve $R^{(\mathcal{I},r)}$ (and hence virtual valuations); or (ii) to define an auction $A^{(\mathcal{I},r)}$. The "switching trick" formalizes the connection between them: the expected maximum virtual welfare with the modified virtual valuations (corresponding to the derivative of $R^{(\mathcal{I},r)}$) equals the expected virtual welfare of the modified auction $A^{(\mathcal{I},r)}$ with the original virtual valuations.

More formally, let $x_i : \mathbb{R}_+^n \to \mathbb{R}_+$ be the ex-post allocation function of the welfare maximizing truthful auction that takes the bids $\mathbf{b}$ of all players and results in the allocation to bidder $i$. The *interim* allocation function $y_i : [0,1] \to \mathbb{R}_+$ is the expected allocation to bidder $i$ when her quantile is $q$, where the expectation is over the quantiles of the other bidders: $y_i(q_i) = \mathbb{E}_{\mathbf{q}_{-i} \sim U[0,1]^{n-1}}[x(F^{-1}(1-q_i), F^{-1}(\mathbf{1} - \mathbf{q}_{-i}))]$ where $F^{-1}(\mathbf{1} - \mathbf{q}_{-i})$ is $\mathbf{b}_{-i}$ for which each $b_j = F(1-q_j)$. For example, in the standard Vickrey (single-item) auction with $n$ bidders, every bidder $i$ has the interim allocation function $y_i(q) = (1-q)^{n-1}$.[10]

For every auction of the form $A^{(\mathcal{I},r)}$, the interim allocation function $y_i^{(\mathcal{I},r)}$ of a bidder $i$ can be expressed in terms of the interim allocation function $y_i$ without ironing and reserve price (see also Figure 2.1b):

$$y_i^{(\mathcal{I},r)}(q) = \begin{cases} 0 & \text{if } q > F(r) \\ \frac{1}{q_b - q_a} \int_{q_a}^{q_b} y(q)dq & \text{if } q \in [q_a, q_b) \text{ with } [F^{-1}(1-q_b), F^{-1}(1-q_a)) \in \mathcal{I} \\ y_i(q) & \text{otherwise.} \end{cases} \tag{2.4}$$

**Proposition 2.3** (Switching Trick)**.** *Consider a matroid or position auction setting, as above. For every valuation distribution $F$, every reserve price $r$, every set $\mathcal{I}$ of disjoint ironing intervals, and every bidder $i$,*

$$\mathbb{E}_{q \sim U[0,1]}[R(q) \cdot (y_i^{(\mathcal{I},r)})'(q)] = \mathbb{E}_{q \sim U[0,1]}[R^{(\mathcal{I},r)}(q) \cdot y_i'(q)].$$

*Proof.* Fix $F$, $\mathcal{I}$, $r$, and $y$. Let $y^{(\mathcal{I},r)}$ be the interim allocation rule from running auction $A^{(\mathcal{I},r)}$. Let $R$ be the revenue curve of $F$ and let $R^{(\mathcal{I},r)}$ denote the revenue curve induced by $\mathcal{I}$ and $r$.

- Define a distribution $F^{(\mathcal{I},r)}$ (which is not equal to $F$ unless $\mathcal{I} = \emptyset$ and $r = 0$) that has the property that its revenue curve $q \cdot F^{(\mathcal{I},r)}(1-q)$ is $R^{(\mathcal{I},r)}$. To see that this is well-defined, observe the following. Any line $\ell$ through the origin only intersects $R^{(\mathcal{I},r)}$ once (if there are point masses in $F$ then a line through

[10] In general matroid settings, different bidders can have different interim allocation functions (even though valuations are i.i.d.).

| Revenue Curve | Description |
|---|---|
| $R$ | $q \cdot F^{-1}(1-q)$ |
| $\widehat{R}_{\min}$ | $q \cdot \widehat{F^{-1}}(1-q-\epsilon)$ |
| $\widehat{R}_{\max}$ | $q \cdot \widehat{F^{-1}}(1-q+\epsilon+\frac{1}{m})$ |

Table 2.1: Overview of notation for revenue curves.

the origin intersects $R$ in a single interval). This means that we can use $R^{(\mathcal{I},r)}$ to construct $F^{(\mathcal{I},r)}$: $F^{(\mathcal{I},r)}(v)$ is the $q$ for which $q \cdot v$ intersects with $R^{(\mathcal{I},r)}(q)$ (if there are any point masses then there will be a range of $q$ for which this is the case; in that case take the largest such $q$). Alternatively, see [Hartline and Roughgarden, 2008] for an explicit formula for $F^{(\mathcal{I},r)}$.

- If we run the same auction $A^{(\mathcal{I},r)}$ on bidders with values drawn from $F^{(\mathcal{I},r)}$, the expected revenue is identical to the auction with bidder values drawn from $F$:

$$\mathbb{E}_{q\sim U[0,1]}[R(q) \cdot \left(y^{(\mathcal{I},r)}\right)'(q)] = \mathbb{E}_{q\sim U[0,1]}[R^{(\mathcal{I},r)}(q) \cdot \left(y^{(\mathcal{I},r)}\right)'(q)].$$

  This can easily be seen by filling in the definitions from (2.3) and (2.4).

- If the bidders have distribution $F^{(\mathcal{I},r)}$, then we might as well not iron or have a reserve price at all; so

$$\mathbb{E}_{q\sim U[0,1]}[R^{(\mathcal{I},r)}(q) \cdot \left(y^{(\mathcal{I},r)}\right)'(q)] = \mathbb{E}_{q\sim U[0,1]}[R^{(\mathcal{I},r)}(q) \cdot y'(q)].$$

  This is also easily seen by filling in the definitions.

□

### 2.2.3 Notation

In the remainder of this paper, our analysis will rely on bounding the difference in revenue of an auction with respect to the optimal auction in terms of their revenue curves. We will use the following conventions, see Table 2.1. The unaltered revenue curve for distribution $F$ is denoted by $R(q) = q \cdot F^{-1}(1-q)$. To denote when we use an estimator for a revenue, i.e. a revenue curve that is constructed based on samples, we use a hat: $\widehat{R}(q) = q \cdot \widehat{F^{-1}}(1-q)$. Based on the available samples we construct high-probability upper and lower bounds for $R$, that are thus denoted as $\widehat{R}_{\max}(q) = q \cdot \widehat{F^{-1}}(1-q+\epsilon+\frac{1}{m})$ and $\widehat{R}_{\min}(q) = q \cdot \widehat{F^{-1}}(1-q-\epsilon)$.

We use a superscript to denote when a revenue curve is ironed and has a reserve price. For a general set of ironing intervals $\mathcal{I}$ and reserve price $r$, $R^{(\mathcal{I},r)}$ is the revenue

curve induced by it, see (2.3). The superscript $\star$ denotes that the revenue curve is optimally ironed and reserved, i.e. $R^\star$ is the revenue curve of Myerson's auction using $F$, and $\widehat{R}^\star_{\max}$ is the revenue curve corresponding to the convex hull of $\widehat{R}_{\max}$ that additionally stays constant after the highest point. Finally, we use $R^{alg}$ and $R^{opt}$ to denote an algorithm ALG's revenue curve and the optimal revenue curve for $F$ respectively (thus $R^{opt} = R^\star$, but when appropriate we use $R^{opt}$ to emphasize its relation to $R^{alg}$).

For the ironing intervals $\mathcal{I}$ (and reserve price $r$) we use $\mathcal{I}_q$ (resp. $r_q$) when it is important that the ironing intervals are defined in *quantile space*. Finally, $\mathcal{I}_{opt}$, $\mathcal{I}_{alg}$, and $\mathcal{I}_{max}$ (and similarly for reserve price $r$) refer to the ironing intervals of the optimal auction, algorithm ALG and the optimal ironing intervals for $\widehat{R}_{\max}$ respectively.

## 2.3 Additive Loss in Revenue for Single-Item Auctions

For ease of exposition, most of the technical sections focus on an unknown distribution with support in $[0, H]$. Section 2.6 explains how our results extend to all distributions for which the optimal auction does not obtain a constant fraction of its expected revenue from valuation profiles with at least one extremely high-valued bidder.

This section describes an algorithm that takes a set $X$ of $m$ samples, and a confidence parameter $\delta$ as input, and outputs a set $\mathcal{I}$ of ironing intervals and a reserve price $r$, both in value space. This section focuses on the case where $\mathcal{I}$ and $r$ are used in a single-item auction $A^{(\mathcal{I},r)}_{(1)} \in \mathcal{A}$ (recall the notation in Section 2.2) and shows that the additive loss in revenue of $A^{(\mathcal{I},r)}_{(1)}$ with respect to the revenue of the optimal auction $A^{opt}_{(1)}$ for single-item auctions is $O(\epsilon \cdot n \cdot H)$, with $\epsilon = \sqrt{\frac{\ln 2\delta^{-1}}{2m}}$. In section 2.4 we extend the results to matroid and position auctions.

**Theorem 2.4** (Main Theorem)**.** *For a single-parameter environment with optimal auction of the form $A^{(\mathcal{I},r)}$ with $n$ i.i.d. bidders with values from unknown irregular distribution $F$, with $m$ i.i.d. samples from $F$, the additive loss in expected revenue of Algorithm 2 compared to the optimal expected revenue is at most $3 \cdot n \cdot H \cdot \sqrt{\frac{\ln 2\delta^{-1}}{2m}}$ with probability at least $1 - \delta$.*

### 2.3.1 The Empirical Myerson Auction

We run a variant of the Empirical Myerson auction, which we have divided into two parts: the first is a learning algorithm *ALG* (Algorithm 1) that computes ironing intervals $\mathcal{I}$ and a reserve price $r$ based on samples $X$ and confidence parameter $\delta$. The

**Algorithm 1** Compute the ironing intervals $\mathcal{I}$ and reserve price $r$.

ComputeAuction$(X, \delta)$

1 Construct $\widehat{F^{-1}}$ from $X$; let $\epsilon = \sqrt{\frac{\ln 2 \cdot \delta^{-1}}{2|X|}}$.
2 Construct $\widehat{R}_{\min}(q) = q \cdot \widehat{F^{-1}}(1 - q - \epsilon)$.
3 Compute the convex hull $\mathcal{CH}(\widehat{R}_{\min})$, of $\widehat{R}_{\min}$.
4 Let $\mathcal{I}_q$ be the set of intervals where $\widehat{R}_{\min}$ and $\mathcal{CH}(\widehat{R}_{\min})$ differ.
5 **for each** quantile ironing interval $(a_i, b_i) \in \mathcal{I}_q$
6 Add $[\widehat{F^{-1}}(1 - b_i - \epsilon),\ \widehat{F^{-1}}(1 - a_i - \epsilon))$ to $\mathcal{I}$.
7 Let the reserve quantile be $r_q = \arg\max_q \widehat{R}_{\min}(q)$.
8 Let the reserve price be $r = \widehat{F^{-1}}(1 - r_q - \epsilon)$.
9 **return** $(\mathcal{I}, r)$

**Algorithm 2** Empirical Myerson.

EmpiricalMyerson$(X, \delta, \mathbf{b})$

1 $\mathcal{I}, r \leftarrow$ ComputeAuction$(X, \delta)$
2 **return** $A_{(1)}^{\mathcal{I},r}(\mathbf{b})$

second step is to run the welfare-maximizing auction subject to ironing and reservation (Algorithm 2). In this section we focus on analyzing the single-item auction, but the only place this is used is in line 2 of Algorithm 2. Auctions for position auctions and matroid environments use the same learning algorithm (Algorithm 1) but then run the welfare-maximizing auction for position auctions or matroid environments respectively; we return to this in Section 2.4.

The Empirical Myerson auction takes an estimator for the quantile function $\widehat{F^{-1}}$ and constructs its revenue curve. From this, the convex hull $\mathcal{CH}(R)$ is computed and wherever $\mathcal{CH}(R)$ and $R$ disagree, an ironing interval is placed. Then, the highest point on $R$ is used to obtain the reserve price quantile $q_r = \arg\max_q R(q)$. Note that this is all done in quantile space, but we need to specify the reserve price and ironing intervals in value space. So the last step is to use the empirical CDF $\widehat{F}$ to obtain the values at which to place the reserve price and ironing intervals.

Our learning algorithm follows that approach, with the exception that in line 2 of ComputeAuction, we take the empirical quantile function to be $\widehat{F^{-1}}(1 - q - \epsilon)$ rather than the arguably more natural choice of $\widehat{F^{-1}}(1 - q)$. The motivation here is

to protect against overestimation — recall the cautionary tale of Example 2.2. From the DKW inequality we can derive that $\widehat{F^{-1}}(1-q-\epsilon) \leq F^{-1}(1-q)$ with probability $1-\delta$ (we prove this in Lemma 2.6), so we would hope that using this will sufficiently protect against overestimation, while incurring only a modest loss in revenue due to underestimation. That this approach indeed leads to good revenue guarantees is shown in the remainder of this section.

### 2.3.2 Additive Revenue Loss in Terms of Revenue Curves

We start with a technical lemma that reduces bounding the loss in revenue to bounding the estimation error due to using samples as opposed to the true distribution $F$.

**Lemma 2.5.** *For a distribution $F$, let $R^{alg}$ be the revenue curve induced by an algorithm $ALG \in \mathcal{A}$ and let $R^{opt}$ be the optimal induced revenue curve. The additive revenue loss of $ALG$ with respect to $OPT$ is at most:*

$$n \cdot \max_{q \in [0,1]} \left( R^{opt}(q) - R^{alg}(q) \right).$$

*Proof.* First, to calculate the ex ante expected revenue of a bidder $i$ with interim allocation function $y_i$,[11] revenue curve $R$, ironing intervals $\mathcal{I}$ and reserve price $r$, we have by [Myerson, 1981]:

$$Rev[R, \mathcal{I}, r] = -\mathbb{E}_{q \sim U[0,1]}[R(q) \cdot (y^{(\mathcal{I},r)})'(q)]. \tag{2.5}$$

Next, we apply the Switching Trick of Proposition 2.3. Let $\mathcal{I}_{alg}, \mathcal{I}_{opt}$ be the sets of ironing intervals and $r_{alg}, r_{opt}$ be the reserve prices of $ALG$ and $OPT$ respectively. This yields total revenues:

$$Rev[F, \mathcal{I}_{alg}, r_{alg}] = \sum_{i=1}^{n} \int_0^1 -R^{alg}(q) y_i'(q) dq,$$
$$Rev[F, \mathcal{I}_{opt}, r_{opt}] = \sum_{i=1}^{n} \int_0^1 -R^{opt}(q) y_i'(q) dq.$$

Note that the interim allocation function $y_i$ for bidder $i$ is the same one in both equations; the only difference between $y_i^{(\mathcal{I}_{opt}, r_{opt})}$ and $y_i^{(\mathcal{I}_{alg}, r_{alg})}$ was the ironing intervals and reserve price, so after applying the switching trick, $y_i$ is simply the welfare-maximizing

[11] For single-item auctions with i.i.d. bidders, all bidders share the same interim allocation function. We write $y_i$ to facilitate our extension to matroid environments in the next section.

interim allocation rule for bidder $i$.[12] This is the key point in our analysis, and it allows us to compare the expected revenue of both auctions directly:

$$\begin{aligned}
Rev[F, \mathcal{I}_{opt}, r_{opt}] - Rev[F, \mathcal{I}_{alg}, r_{alg}] &= \sum_{i=1}^{n} \left( - \int_0^1 R^{opt}(q) y_i'(q) dq + \int_0^1 R^{alg}(q) y_i'(q) dq \right) \\
&= \sum_{i=1}^{n} \int_0^1 \left( R^{opt}(q) - R^{alg}(q) \right) \left( -y_i'(q) \right) dq \\
&\leq \max_{q \in [0,1]} \left( R^{opt}(q) - R^{alg}(q) \right) \cdot \sum_{i=1}^{n} \int_0^1 -y_i'(q) dq \\
&= \max_{q \in [0,1]} \left( R^{opt}(q) - R^{alg}(q) \right) \cdot \sum_{i=1}^{n} \left( -y_i(1) + y_i(0) \right) \\
&\leq n \cdot \max_{q \in [0,1]} \left( R^{opt}(q) - R^{alg}(q) \right).
\end{aligned}$$

The inequality holds as $-y_i'$ is non-negative. The last inequality holds because $y_i$ always lies between 0 and 1. Rearranging the terms yields the claim. □

This is significant progress: the additive loss in revenue can be bounded in terms of the induced revenue curves of an algorithm ALG and the optimal algorithm, two objects that we have some hope of getting a handle on. Of course, we still need to show that the ironed revenue curve of Algorithm 1 is pointwise close to the ironed revenue curve induced by the optimal auction (Section 2.3.3).

### 2.3.3 Bounding the Error in the Revenue Curve

We implement the following steps to prove that the error in the learning algorithm's estimation of the revenue curve is small. The proofs of this section appear in the full version.

- (Lemma 2.6) We show that we can sandwich the actual revenue curve (without ironing or reserve price) $R$ between two empirical revenue curves, $\widehat{R}_{\min}$ and $\widehat{R}_{\max}$ that are defined using the empirical quantile function.
- (Lemma 2.7 and Lemma 2.8) Let $\widehat{R}^{\star}_{\max}$ (resp. $\widehat{R}^{\star}_{\min}$) be the optimally induced revenue curve for $\widehat{R}_{\max}$ (resp. $\widehat{R}_{\min}$). The revenue curve induced by Algorithm 1, $R^{alg}$, is pointwise higher than the optimal induced revenue curve of the lower

[12] E.g., for single-item auctions, this is just the probability that bidder $i$ has the largest valuation (given its quantile).

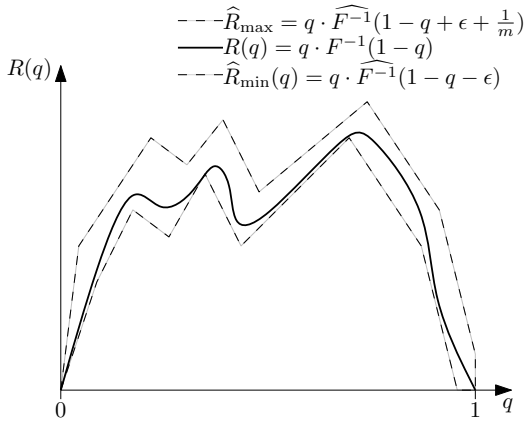



Figure 2.2: We can sandwich $R$ between two estimated revenue curves $\widehat{R}_{\min}$ and $\widehat{R}_{\max}$.

bound $\widehat{R}^{\star}_{\min}$, and the optimal induced revenue curve for the upper bound, $\widehat{R}^{\star}_{\max}$, is pointwise higher than $R^{opt}$.

- (Lemma 2.10) Finally, we show that $\widehat{R}^{\star}_{\max}(q) - \widehat{R}^{\star}_{\min}(q)$ is small for all $q$, and therefore the additive loss is small.

**Lemma 2.6.** *For a distribution $F$ and $m$ samples from $F$. Let $\widehat{R}_{min}(q) = q \cdot \widehat{F^{-1}}(1 - q - \epsilon)$ and $\widehat{R}_{max}(q) = q \cdot \widehat{F^{-1}}(1 - q + \epsilon + \frac{1}{m})$. With probability at least $1 - \delta$ for all $q \in [0, 1]$:*

$$\widehat{R}_{min}(q) \quad \leq \quad R(q) \quad \leq \quad \widehat{R}_{max}(q).$$

*Proof.* See Figure 2.2 for graphical intuition. We start by proving the first inequality: $\widehat{R}_{\min}(q) \leq R(q)$. By the DKW inequality with probability $1 - \delta$ the following holds for all $v$: $-\epsilon \leq \widehat{F}(v) - F(v)$. Define $q' = F(v)$ to obtain $q' - \epsilon \leq \widehat{F}(v)$. Since $\widehat{F^{-1}}$ is monotonically non-decreasing, we have

$$\widehat{F^{-1}}\,(q' - \epsilon) \;\leq\; \widehat{F^{-1}}\left(\widehat{F}(v)\right) \;\leq\; v \;=\; F^{-1}(q').$$

where the second inequality follows from (2.2) and the equality follows from the definition of $q'$. Finally, let $q = 1 - q'$, and multiply both sides by $q$ to obtain: $\widehat{R}_{\min}(q) \leq R(q)$.

The proof for the upper bound of $R$ is analogous with the exception that we pick up another $\frac{1}{m}$ term to invoke (2.2). □

So while the the algorithm does not know the exact revenue curve $R$, it can be upper bounded by $\widehat{R}_{\max}$ and lower bounded by $\widehat{R}_{\min}$. We'll use $\widehat{R}_{\min}$ to give a lower

bound on the revenue curve $R^{alg}$ induced by Algorithm 1, and $\widehat{R}_{\max}$ to give an upper bound on the revenue curve $R^{opt}$ induced by Myerson's optimal auction. We start with the latter.

**Lemma 2.7.** *Let $R^{opt}$ be the optimal induced revenue curve of $R$, and let $\widehat{R}^{\star}_{max}$ be the optimal induced revenue curve for $\widehat{R}_{max}$. Then with probability $1-\delta$ for all $q \in [0,1]$:*

$$R^{opt}(q) \quad \leq \quad \widehat{R}^{\star}_{max}(q).$$

*Proof.* By Lemma 2.6 we know that for every $q$ with probability $1-\delta$, $\widehat{R}_{\max}(q) \geq R(q)$. First take the effect of optimally ironing both curves into account. Optimal ironing will lead to induced revenue curves $\widehat{R}^{(\mathcal{I}_{max})}_{\max}$ and $R^{(\mathcal{I}_{opt})}$ which are their convex hulls. Since $\widehat{R}_{\max}$ is pointwise higher, the convex hull $\widehat{R}^{(\mathcal{I}_{max})}_{\max}$ is pointwise higher than $R^{(\mathcal{I}_{opt})}$, the convex hull of $R$. Thus, for all $q \in [0,1]$, we have $\widehat{R}^{(\mathcal{I}_{max})}_{\max}(q) \geq R^{(\mathcal{I}_{opt})}(q)$.

What remains to be proven is that this property is maintained after the effect of the optimal reserve price of the curve. Let $q$ be the reserve quantile for $R$; we distinguish between two cases. For all $q' \in [0,q]$, we know that $\widehat{R}^{\star}_{\max}(q') \geq R^{\star}(q')$, because $R^{\star}$ is the same on this interval as $R^{(\mathcal{I}_{opt})}$, and setting the reserve price for $\widehat{R}_{\max}$ can only pointwise increase the curve. For all points $q' \in [q,1]$ we have that $R^{\star}$ is constant with value $R^{\star}(q)$. Recall that $\widehat{R}^{(\mathcal{I}_{max})}_{\max}(q) \geq R^{(\mathcal{I}_{opt})}(q)$ and since $\widehat{R}^{\star}_{\max}$ is monotonically non-decreasing, we have that $\widehat{R}^{\star}_{\max}(q') \geq \widehat{R}^{\star}_{\max}(q) \geq R^{\star}(q) = R^{\star}(q')$. □

So with high probability $\widehat{R}^{\star}_{\max}$ is pointwise higher than $R^{opt}$. Proving that $\widehat{R}^{\star}_{\min}$ is a lower bound for $R^{alg}$ is slightly more involved since the ironing intervals and reserve price are given by Algorithm 1, and may not be optimal. Therefore, the induced revenue curve $R^{alg}$ is in general not concave, and the reserve quantile may not be at the highest point of the curve. In the following lemma, we use the fact that that the ironing intervals and reserve price of $R^{alg}$ were chosen based on $\widehat{R}_{\min}$.

**Lemma 2.8.** *Let $R^{alg}$ be the revenue curve induced by Algorithm 1 and let $\widehat{R}^{\star}_{min}$ be the optimal induced revenue curve for $\widehat{R}_{min}$. Then with probability $1-\delta$ for all $q \in [0,1]$:*

$$\widehat{R}^{\star}_{min}(q) \quad \leq \quad R^{alg}(q).$$

First we show that to prove pointwise dominance of curve $R^{alg}$ over $\widehat{R}_{\min}$, it is sufficient to show that any ray from the origin that intersects the revenue curves, first intersects with $\widehat{R}_{\min}$ and then $R^{alg}$.

**Proposition 2.9** (Ray Dominance)**.** *For two (potentially ironed) revenue curves $R$ and $R'$, if all rays from the origin that intersect with $R$ or $R'$ intersect $R$ before $R'$, then it must be that $R(q) \leq R'(q)$ for $q \in [0,1]$.*

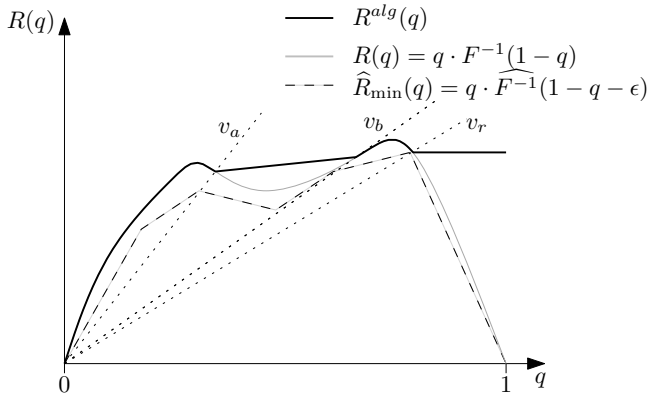



Figure 2.3: When we pick a value range to iron based on $\widehat{R}_{\min}$, its effect on the actual revenue curve can be seen. The quantiles of the start and end point of the ironing procedure are given by the line that intersects the start and end point on $\widehat{R}_{\min}$ and the origin.

*Proof.* Fix $R, R'$. WLOG assume that $R$ and $R'$ are not ironed.[13] First observe that a revenue curve $S$ consists of the set of points $\{(1 - F(v), v \cdot (1 - F(v))) : v \in \mathbb{R}^+\}$, where $F$ is the CDF corresponding to $S$. Therefore a ray from the origin with slope $v'$ intersects $S$ on the set of points $\{(1 - F(v), v \cdot (1 - F(v))) : 1 - F(v) = 1 - F(v')\}$. Since $1 - F(v)$ is non-increasing, this can only happen in either a single point, or single line segment.

We prove the contrapositive. If $R$ is not dominated by $R'$, then by continuity there is an interval $(q_1, q_2)$ with $q_1 < q_2$ where $R$ is above the graph of $R'$. Let $q' = \frac{q_1+q_2}{2}$ be the midpoint. Take the ray that starts at the origin and has slope $\frac{R'(q')}{q'}$. This ray intersects $R'$ in the point $(q', R'(q'))$. Since $R'(q) > R(q)$ for $q \in (q_1, q_2)$, and $R'$ is continuous, the ray must intersect with $R'$ after it intersected with $R$. Since rays from the origin only intersect a revenue curve once, it could not have intersected $R'$ before $R$ either. □

What remains to be shown is that all rays through the origin intersect $\widehat{R}_{\min}$ before they intersect $R^{alg}$.

*Proof of Lemma 2.8.* Again we first argue that after ironing the induced curve $\widehat{R}_{\min}^{(\mathcal{I}_{min})}$ is completely below $R^{(\mathcal{I}_{alg})}$, and subsequently show that this is after setting the reserve price $\widehat{R}^{\star}_{\min}$ is below $R^{alg}$.

Algorithm 1 picks the ironing intervals based on where $\widehat{R}^{\star}_{\min}$ differs from $\mathcal{CH}(\widehat{R}_{\min})$, so there is a matching between the ironing intervals in $\widehat{R}_{\min}$ and $R^{alg}$. However, $\widehat{R}_{\min}$

[13] Recall that from the Switching Trick, any ironed revenue curve can be written as the revenue curve for an unironed, different CDF.

and $R^{alg}$ are not ironed on the same intervals in quantile space, but on the same intervals in *value* space. We treat the effect of ironing, on an interval-by-interval basis; each time ironing both $\widehat{R}_{\min}$ and $R$ with corresponding ironing intervals.[14]

Initially, with probability $1-\delta$ we have that $\widehat{R}_{\min} \leq R$ for all $q \in [0,1]$. If we start out with $\widehat{R}_{\min} \leq R$ and we iron one interval $[v_a, v_b]$, then afterwards $\widehat{R}_{\min}$ will still be at most $R$ everywhere: both $\widehat{R}_{\min}$ and $R$ are ironed on the same interval in value space $[v_a, v_b)$. We can find the quantiles between which $\widehat{R}_{\min}$ and $R$ are ironed, by looking at the line with slope $v_a$ (resp. $v_b$) that goes through the origin, see Figure 2.3. Anything counterclockwise from the $v_a$ line, and clockwise from the $v_b$ line is unchanged, and therefore $R$ will be higher than $\widehat{R}_{\min}$. Between the lines for $v_a$ and $v_b$ the curves $\widehat{R}_{\min}$ and $R$ are replaced by a line segment. Since the latter's endpoints are above $\widehat{R}_{\min}$, the entire line segment of $R$ is above $\widehat{R}_{\min}$. Therefore, after ironing this interval, the induced $\widehat{R}_{\min}$ is still below the induced $R$. Repeating this for all ironing interval yields that for all $q \in [0,1]$ we have $\widehat{R}_{\min}^{(\mathcal{I}_{min})}(q) \leq R^{(\mathcal{I}_{alg})}(q)$.

Finally, for the reserve price $v_r$, the same analysis holds: the quantiles where the revenue curves are reserved can be found by the intersection with the line with slope $v_r$ that goes through the origin. Everything counterclockwise stays the same, on the clockwise side $\widehat{R}_{\min}$ stays constant at a value that must be at most the value at which $R$ stays constant. Hence for all $q \in [0,1]$ we have $\widehat{R}^{\star}_{\min}(q) \leq R^{alg}(q)$. □

We now have our upper bound and lower bounds in terms of $\widehat{F^{-1}}$. Finally we show that the difference between the two is small.

**Lemma 2.10.** *For a distribution $F$, let $R^{alg}$ be the revenue curve induced by Algorithm 1 and $R^{opt}$ the optimal induced revenue curve. Using $m$ samples from $F$, with probability $1-\delta$ and $\epsilon = \sqrt{\frac{\ln 2\cdot\delta^{-1}}{2m}}$ for all $q$:*

$$R^{opt}(q) - R^{alg}(q) \leq \widehat{R}^{\star}_{max}(q) - \widehat{R}^{\star}_{min}(q) \leq \left(2\epsilon + \frac{1}{m}\right) H \leq 3\epsilon \cdot H.$$

*Proof.* Let $\mathcal{I}_{\max}$ and $r_{\max}$ be the set of ironing intervals and reserve price in quantile space of $\widehat{R}^{\star}_{\max}$. Comparing $\widehat{R}^{\star}_{\max}$ and $\widehat{R}^{\star}_{\min}$ directly is difficult since their ironing intervals and reserve price may be quite different. Instead, we will take the set $\mathcal{I}_{\max}$ of ironing intervals $[a_i, b_i)$, and reserve quantile $r_{\max}$ from $\widehat{R}^{\star}_{\max}$ and use this to iron $\widehat{R}_{\min}(q)$ at intervals $[a_i - 2\epsilon - \frac{1}{m}, b_i - 2\epsilon - \frac{1}{m})$, and set a reserve price of $r_{\max} - 2\epsilon - \frac{1}{m}$.

If we compare $\widehat{R}^{\star}_{\max}$ against this induced curve $\widehat{R}^{(\mathcal{I}_{\max}, r_{\max})}_{\min}$, we have an upper bound for the difference with respect to $\widehat{R}^{\star}_{\min}$, since $\widehat{R}^{\star}_{\min}$ is the optimal induced revenue curve and therefore pointwise higher than $\widehat{R}^{(\mathcal{I}_{\max}, r_{\max})}_{\min}$. We can reason about

[14] This can be made rigorous by induction on the set of ironing intervals.

ironing in quantile space without loss of generality since both $\widehat{R}_{\max}$ and $\widehat{R}_{\min}$ use the same function $\widehat{F^{-1}}$.[15]

There are 3 cases to handle: 1) if $q$ falls in an ironing interval $[a_i, b_i) \in \mathcal{I}_{\max}$, 2) if the quantile $q$ is higher than the reserve quantile $q \geq r_{\max}$ of $\widehat{R}_{\max}$, and 3) when neither of those cases apply. We start with the case for a $q$ where $\widehat{R}^{\star}_{\max}(q) = \widehat{R}_{\max}$, i.e. $q$ does not fall in an ironing interval and is smaller than the reserve quantile:

$$\begin{aligned}
\widehat{R}^{\star}_{\max}(q) &= q \cdot \widehat{F^{-1}}\left(1 - q + \epsilon + \frac{1}{m}\right) \\
&\leq \left(q - 2\epsilon - \frac{1}{m}\right) \cdot \widehat{F^{-1}}\left(1 - q + \epsilon + \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&= \widehat{R}_{\min}\left(q - 2\epsilon - \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&= \widehat{R}^{\star}_{\min}\left(q - 2\epsilon - \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&\leq \widehat{R}^{\star}_{\min}(q) + \left(2\epsilon + \frac{1}{m}\right) H
\end{aligned}$$

where the first inequality holds because $\widehat{F^{-1}}(q) \leq H$ for all $q$ and the last inequality follows because $\widehat{R}^{\star}_{\min}$ is monotonically non-decreasing. Rearranging yields the claim for $q$ not in an ironing interval or reserved.

The other cases follow similarly: consider the case where $q \geq r_{\max}$.

$$\begin{aligned}
\widehat{R}^{\star}_{\max}(q) &= \widehat{R}_{\max}\left(r_{\max}\right) \\
&= r_{\max} \cdot \widehat{F^{-1}}\left(1 - r_{\max} + \epsilon + \frac{1}{m}\right) \\
&\leq \left(r_{\max} - 2\epsilon - \frac{1}{m}\right) \cdot \widehat{F^{-1}}\left(1 - r_{\max} + \epsilon + \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&= \widehat{R}_{\min}\left(r_{\max} - 2\epsilon - \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&\leq \widehat{R}^{\star}_{\min}\left(r_{\max} - 2\epsilon - \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H \\
&\leq \widehat{R}^{\star}_{\min}\left(q\right) + \left(2\epsilon + \frac{1}{m}\right) H
\end{aligned}$$

[15] Alternatively think of it as ironing in value space based on $\widehat{R}_{\max}$, the line through the origin and $\widehat{R}_{\max}(q)$ intersects $\widehat{R}_{\min}$ at $q - 2\epsilon - \frac{1}{m}$.

where the last inequality holds because $r_{\max} - 2\epsilon - \frac{1}{m} \leq r_{\max} \leq q$ and $\widehat{R}^{\star}_{\min}$ is non-decreasing.

Finally the case for when $q$ falls in an ironing interval is analogous: $\widehat{R}^{\star}_{\max}(q)$ is a convex combination of the end points of the ironing interval: $\widehat{R}^{\star}_{\max}(q) = \frac{q-a_i}{b_i-a_i} \cdot \widehat{R}_{\max}(a_i) + \left(1 - \frac{q-a_i}{b_i-a_i}\right) \widehat{R}_{\max}(b_i)$ and hence

$$\widehat{R}^{\star}_{\min}(q) \geq \frac{q-a_i}{b_i-a_i} \cdot \widehat{R}_{\min}\left(a_i - 2\epsilon - \frac{1}{m}\right) + \left(1 - \frac{q-a_i}{b_i-a_i}\right) \widehat{R}_{\min}\left(b_i - 2\epsilon - \frac{1}{m}\right) + \left(2\epsilon + \frac{1}{m}\right) H.$$

And so $\widehat{R}^{\star}_{\max}(q) - \widehat{R}^{\star}_{\min}(q) \leq \left(2\epsilon + \frac{1}{m}\right) H$ in all cases. □

Theorem 2.4 now follows by combining Lemmas 2.5 and 2.10. The additive loss in expected revenue of Algorithm 2 is at most $3\epsilon \cdot n \cdot H$.

*Proof of Theorem 2.4.* By Lemma 2.5 we can express the total additive error of the expected revenue of an algorithm that yields ironing intervals $\mathcal{I}_{alg}$ and reserve price $r_{alg}$ with respect to the optimal auction as:

$$\frac{Rev[F, \mathcal{I}_{opt}, r_{opt}] - Rev[F, \mathcal{I}_{alg}, r_{alg}]}{n} \leq \max_{q \in [0,1]} \left(R^{opt}(q) - R^{alg}(q)\right).$$

By Lemma 2.10, Algorithm 2 yields

$$\max_{q \in [0,1]} \left(R^{opt}(q) - R^{alg}(q)\right) \leq \left(2\epsilon + \frac{1}{m}\right) H.$$

The theorem follows. □

When the optimal revenue is bounded away from zero, we get an analogous sample complexity bound for learning (efficiently) a $(1-\epsilon)$-(multiplicative) approximate auction.

## 2.4 Matroid and Position Environments

The results of the previous section extend to matroid and position auction environments.

**Theorem 2.11.** *For position and matroid auctions with $n$ i.i.d. bidders with values from unknown distribution $F$, $m$ i.i.d. samples from $F$, with probability $1-\delta$, the*

*additive loss in expected revenue of running the welfare-maximizing auction using ironing intervals and reserve price from Algorithm 1 compared to the optimal expected revenue is at most* $3 \cdot n \cdot H \cdot \sqrt{\frac{\ln 2\delta^{-1}}{2m}}$.

The learning algorithm uses the same subroutine (Algorithm 1) to learn ironed intervals and a reserve price, and returns the auction that first deletes all bidders not meeting the reserve and then chooses the feasible outcome maximizing the ironed virtual welfare. The proof follows from Proposition 2.12 and Proposition 2.13, which show that the optimal auctions for matroid and position auction environments are in $\mathcal{A}$ (i.e., have the form $A^{(\mathcal{I},r)}$ for a suitable choice of ironed intervals $\mathcal{I}$ and reserve price $r$), and from Lemmas 2.5–2.10, which rely only on this property.

### 2.4.1 Position Auctions

A position auction [Varian, 2007] is one where the winners are given a position, and position $i$ comes with a certain quantity $x_i$ of the good. The canonical example is that of ad slot auctions for sponsored search, where the best slot has the highest click-through-rate, and subsequent slots have lower and lower click-through-rates. In an optimal auction, the bidder with the highest ironed virtual value gets the best slot, the second highest ironed virtual value the second slot, and so on.

**Proposition 2.12.** *The optimal auction* $A^{opt}_{(pos)}$ *for position auctions can be expressed as an auction with ironing and reserve price in value space:* $A^{opt}_{(pos)} \in \mathcal{A}$.

*Proof.* In the optimal auction, the bidder with the highest ironed virtual value is awarded the first position (with allocation $x_1$), the bidder with the second highest ironed virtual value the second position with $x_2$, and so on. Since the ironed virtual value is monotonically non-decreasing in the value of a bidder, and identical in ironing intervals, this can equivalently be described by an auction in $\mathcal{A}$. □

### 2.4.2 Matroid Environments

In a matroid environment, the feasible allocations are given by matroid $\mathcal{M} = (E, I)$, where $E$ are the players and $I$ are independent sets. The auction can simultaneously serve only sets $S$ of players that form an independent set of the matroid $S \in I$. A special case of this is the rank $k$ uniform matroid, which accepts all subsets of size at most $k$, i.e. it is a $k$-unit auction environment.

In matroid environments, the ex-post allocation function $x_i(\mathbf{b})$ and interim allocation function $y_i(q)$ are no longer the same for each player, e.g. imagine a player $i$ who is not part of any independent set, then $y(q) = 0$ everywhere. However, the optimal allocation can still be expressed in terms of an auction $A \in \mathcal{A}$.

**Algorithm 3** A no-regret algorithm for optimal auctions.

NO-REGRET-AUCTION$(\delta, T)$
1 $\triangleright$ Round 0:
2 Collect a set of bids $\mathbf{b}$, run an arbitrary mechanism
3 $X \leftarrow \mathbf{b}$
4 **for** round $t = 1...T$
5 Collect a set of bids $\mathbf{b}$
6 EMPIRICALMYERSON$(X, \delta/T, \mathbf{b})$
7 $X \leftarrow X \cup \mathbf{b}$

**Proposition 2.13.** *The optimal auction $A^{opt}_{(mat)}$ for matroid auctions can be expressed as an auction with ironing and reserve price in value space: $A^{opt}_{(mat)} \in \mathcal{A}$.*

*Proof.* A property of matroids is that the following simple greedy algorithm yields the optimal solution:

GREEDY$(E, I)$
1 $S \leftarrow \emptyset$
2 **while** $\{i : i \notin S \wedge S \cup \{i\} \in I\} \neq \emptyset$
3 add $\arg\max\{v_i : i \notin S \wedge S \cup \{i\} \in I\}$ to $S$
4 **return** $S$

where $v_i$ is the (ironed virtual) value associated with bidder $i$. Since the order of largest values is the same for both virtual values and bids (up to ties), the allocation of the optimal auction is identical to the auction that irons on $\mathcal{I}_{opt}$ and has reserve price $r_{opt}$ (up to tie-braking); hence $A^{opt}_{(mat)} \in \mathcal{A}$. □

## 2.5 No-Regret Algorithm

So far we assumed access to a batch of samples before having to choose an auction. In this section we show that running the algorithm in a repeated setting, using past bidding behavior as the samples, leads to a no regret algorithm. The goal here is to achieve total additive error $o(T) \cdot O(\text{poly}(n, H, \delta))$ — the error can be polynomial in all parameters except the time horizon $T$, for which it should be sublinear. We show that for Algorithm 3 the total loss grows as $\widetilde{O}(\sqrt{T}\sqrt{n}\sqrt{\log(\delta^{-1})}H)$ and hence results in a no-regret algorithm.

We run Algorithm 3. Invoking Theorem 2.4 with confidence parameter $\delta/T$ and taking a union bound over the rounds, we have the following fact.

**Proposition 2.14.** *With probability $1-\delta$, for all rounds simultaneously, each round $t \in [1,T]$ of Algorithm 3 has additive loss at most $3\sqrt{\frac{\ln(2T\delta^{-1})}{2nt}} \cdot n \cdot H$.*

This leads to the following no-regret bound.

**Theorem 2.15.** *With probability $1-\delta$, the total additive loss of Algorithm 3 is $O(\sqrt{n}\sqrt{T\log T}\sqrt{\log \delta^{-1}} \cdot H)$, which is $\widetilde{O}(T^{1/2})$ with respect to $T$.*

*Proof.* By Proposition 2.14 with probability $1-\delta$ for all rounds simultaneously, the additive loss for round $t$ is bounded by $3\sqrt{\frac{\ln(2T\delta^{-1})}{2nt}} \cdot n \cdot H$. The loss of day 0 is at most $H \cdot n$. The total loss can then be bounded by:

$$(n \cdot H) \cdot \left(1 + 3\sum_{t=1}^{T} \sqrt{\frac{\ln(2T\delta^{-1})}{2nt}}\right)$$

We can rewrite the sum:

$$\begin{aligned}\sum_{t=1}^{T} \sqrt{\frac{\ln(2T/\delta)}{2nt}} &= \sqrt{\frac{\ln(2T/\delta)}{2n}} \cdot \sum_{t=1}^{T} \sqrt{\frac{1}{t}} \\ &\leq \sqrt{\frac{\ln(2T/\delta)}{2n}} \cdot 2\sqrt{T} \\ &= \sqrt{\frac{2T\ln(2T/\delta)}{n}}\end{aligned}$$

Hence the total loss is

$$(n \cdot H) \cdot \left(1 + 3\sqrt{\frac{2T\ln(4T/\delta)}{n}}\right) = O(\sqrt{n}\sqrt{T\log T}\sqrt{\log \delta^{-1}} \cdot H);$$

the dependence on $T$ is $O(\sqrt{T\log T}) = \widetilde{O}(\sqrt{T})$. □

The bound of $O(\sqrt{T\log T})$ is almost tight, as there is a lower bound of $\Omega(\sqrt{T})$ given by Cesa-Bianchi et al. [2015]. Also note that if we do not know $T$ a priori, we can use a standard doubling argument to obtain the same asymptotic guarantee.

## 2.6 Unbounded Distributions

Our results do not require bounded valuations. As mentioned in the introduction (Footnote 4), however, for any non-trivial results it is necessary to make some type of assumption to exclude certain pathological distributions. Our approach is to limit how

much revenue the optimal auction obtains from the tail of the distribution. Formally, for a parameter $H$ and function $\eta(n)$, we say that $F$ is an $(H, \eta(n))$ *distribution* if, with $n$ bidders with valuations drawn i.i.d. from $F$, the contribution to the optimal expected revenue by valuation profiles with some bidder with valuation more than $H$ is at most $\eta(n)$. See below for a concrete example.

Let $ALG$ be the algorithm that rounds down any bid bigger than $H$ to $H$, and then runs Algorithm 2 on the rounded bids. We get the following result.

**Theorem 2.16.** *If $F$ is an $(H, \eta(n))$ distribution and the number $m$ of samples is $\Omega\left(\epsilon^{-2} \ln \delta^{-1} n^2 H^2\right)$, then with probability $1-\delta$ we have $Rev[ALG] \geq (1-\epsilon)Rev[OPT] - \eta(n)$.*

*Proof.* Split $Rev[OPT]$ into the revenue that is generated when 1) there is at least one bidder with value strictly more than $H$, and 2) when all bidders have value at most $H$. Let $Z$ be the event that there is at least one bidder with value strictly more than $H$. Using Theorem 2.4 and the definition of an $(H, \eta(n))$-distribution, we have

$$\begin{aligned} Rev[ALG] &\geq (1-\epsilon) \cdot Rev[OPT|\neg Z] \\ &\geq (1-\epsilon) \cdot (Rev[OPT] - \eta(n)) \\ &\geq (1-\epsilon) \cdot Rev[OPT] - \eta(n). \end{aligned}$$

□

Many distributions, including many irregular and unbounded distributions, are $(H, \eta(n))$-distributions with reasonable parameters. We give one example to prove this point; it should be clear from the exercise that there are many other examples.

**Example 2.17** (Unbounded Irregular Distribution)**.** *Take the distribution*

$$F(x) = \begin{cases} x(1-e^{-1}) & \text{for } 0 \leq x \leq 1 \\ 1 - e^{-x} & \text{for } x \geq 1. \end{cases}$$

*This distribution is with probability $1-e^{-1}$ uniform on $[0,1]$ and otherwise exponential with rate 1. The revenue curve (Figure 2.4) is not concave, so this distribution is not regular. Elementary computations show that it is a $(10 \ln n, 10^{-6})$-distribution (for all $n$). Hence with $\Omega\left(\epsilon^{-2} \ln \delta^{-1} n^2 \ln^2 n\right)$ samples, $Rev[ALG] \geq (1-\epsilon)Rev[OPT] - 10^{-6}$ (with probability at least $1-\delta$). Since $Rev[OPT] \approx 0.2$ even with $n = 1$, this additive error is negligible.*

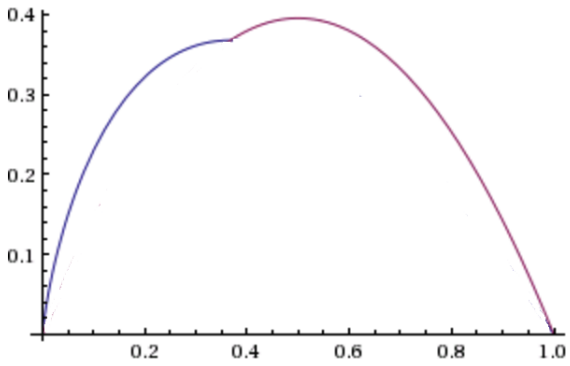



Figure 2.4: The revenue curve of distribution $F$ in Example 2.17 that's both irregular and unbounded.

# Chapter Appendix

## 2.A Reduced Information Model

This appendix considers a reduced information model for single item auctions with i.i.d. bidders. The information that the auctioneer sees is the second highest bid, see also [Cesa-Bianchi et al., 2015]. The goal is to model an observer who can see the outcome of past auctions, and perhaps submit a "shill bid" to set a reserve, but cannot directly observe the bids.

We assume we observe $m$ samples from the second highest order statistic (so out of $n$ i.i.d. bids, we see the second highest bid). We show that there are distributions such that, in order to get the performance close to that of running Myerson in the reduced information model, you first need to see at least an exponential number of samples. This is far more than what would be needed with regular valuation distributions, where only the monopoly price is relevant.

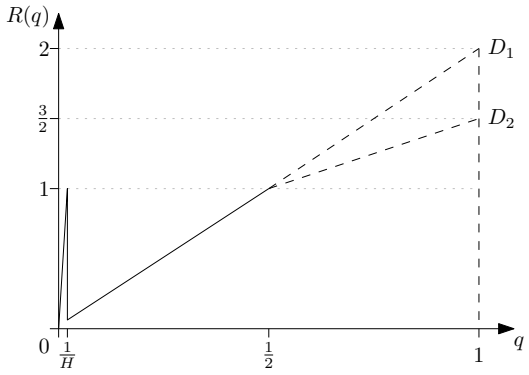
### 2.A.1 Lower Bound

**Theorem 2.18.** *For any $\epsilon > 0$, to obtain $\frac{1}{4} - \epsilon$ additive loss compared to Myerson's auction, with constant probability you need $\Omega(2^n/n)$ samples from $F_{(2)}$.*

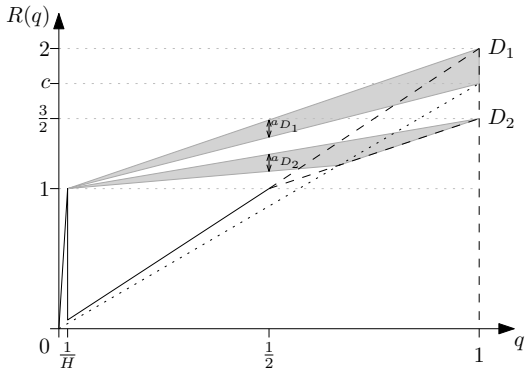
Before proving the statement, let's think about what this means: it means that if all we observe are samples from $F_{(2)}$, for large $n$, we cannot hope to get a vanishing regret in a polynomial (in $n$) number of steps. Moreover, on this distribution, Myerson obtains at most 2 expected revenue, so we lose at least $\frac{1}{8}$ of the profit (see corollary after the proof).

*Proof of Theorem 2.18.* We'll give 2 distributions, $D_1$ and $D_2$, such that we need $\Omega(2^n)$ samples from $F_{(2)}$ to differentiate between the two. Moreover, we'll show that every auction that approximates the optimal auction for either $D_1$ or $D_2$, incurs an additive loss of at least $\frac{1}{2}(1 - \frac{1}{\sqrt{2}})$ on one of the two.

We define $D_1$ and $D_2$ by their quantile function:




(a) Two distributions, $D_1$ and $D_2$ that are hard to distinguish using samples from $F_{(2)}$. The distributions agree on $q \in [0, 1/2]$ and disagree elsewhere.




(b) Consider ironing on $[c, H)$. The top shaded area indicates the loss with respect to $D_1$ and the bottom shaded area represents the loss with respect to $D_2$.

Figure 2.5: Lower bound example for optimal auctions using samples from $F_{(2)}$.

$$D_1^{-1}(q) = \begin{cases} H & \text{for } q \leq \frac{1}{H} \\ 2 & \text{otherwise} \end{cases}$$

$$D_2^{-1}(q) = \begin{cases} H & \text{for } q \leq \frac{1}{H} \\ 2 & \text{for } \frac{1}{H} < q \leq \frac{1}{2} \\ 1 + \frac{1}{2q} & \text{otherwise.} \end{cases}$$

Here $H \gg n$ is a sufficiently large constant. The two distributions agree on $q \in [0, \frac{1}{2}]$. For $q \in (\frac{1}{2}, 1]$ they differ, and the effect on the revenue curve can be seen in Figure 2.5a.

To complete the proof we need to show 2 things: 1) that differentiating between $D_1$ and $D_2$ based on samples from $F_{(2)}$ requires $\Omega(2^n/2)$ samples, and 2) that no auction can simultaneously approximate the optimal solution for both.

The first aspect of this is straightforward. Whenever we observe a sample from $F_{(2)}$ from the top half quantile: $q \in [0, \frac{1}{2}]$, we get no information, since this is the same for both distributions. So a necessary condition for differentiating between $D_1$ and $D_2$ is if we observe a sample from $q \in (\frac{1}{2}, 1]$. For this to happen, we need that out of $n$ draws, $n-1$ draws are in the bottom quantile. This happens with probability $\frac{n}{2^{n-1}}$, so after $\frac{2^n}{n}$ samples, with probability approximately $1 - \frac{1}{e}$ we haven't seen a sample that will differentiate the two distributions.

Now we will show that until the moment when you can differentiate between $D_1$ and $D_2$, there is no way to run an auction that performs well for both. Note that

the optimal auction for $D_1$ is to iron $[2, H)$ and the optimal auction for $D_2$ is to iron $[\frac{3}{2}, H)$. Neither has a reserve price. The set of all auctions that could potentially work well for either, is the set of auctions that irons $[c, H)$ (see Figure 2.5b).

For simplicity, we'll only count the loss we incur in the range $[0, \frac{1}{2}]$, take $H \to \infty$ and we see that if $a$ is the height of the shaded region (of $D_1$ resp. $D_2$) at $q = \frac{1}{2}$, then its loss with respect to the optimal auction is:

$$\begin{aligned}
n \cdot \int_0^{\frac{1}{2}} 2aq \cdot (n-1)(1-q)^{n-2} dq &= \left[2anq(1-q)^{n-1}\right]_0^{1/2} - 2an \int_0^{\frac{1}{2}} (1-q)^{n-1} dq \\
&= 2an \left(\frac{1}{2}\right)^{n-1} - 2an \left[\frac{1}{n}(1-q)^n\right]_0^{1/2} \\
&= 2an \left(\frac{1}{2}\right)^{n-1} - 2an \frac{1}{n} \left(\frac{1}{2}\right)^n + 2an \frac{1}{n} \\
&= 2an \left(\frac{1}{2}\right)^{n-1} \left(1 - \frac{1}{2n}\right) + 2a \\
&\geq 2a
\end{aligned}$$

This means, that if we can show that for all choices of $c \in [3/2, 1]$ at least one of $D_1, D_2$ has a large $a$, we are done. The closer $c$ is to 2, the smaller $a_{D_2}$ is, the closer $c$ is to $3/2$, the smaller $a_{D_1}$, so we'll balance the two and show that neither is small enough.

Finding $a_{D_1}$ in terms of $c$ is easy enough: $a_{D_1} = \frac{2-c}{2}$. $a_{D_2}$ is slightly harder: we first need to find the intersection of $q \cdot c$ and $R_{D_2}(q)$:

$$\begin{aligned}
\frac{1}{2} + q &= qc \\
(c-1)q &= \frac{1}{2} \\
q &= \frac{1}{2(c-1)}
\end{aligned}$$

So the lower part of the shaded area of $D_2$ is a line that passes through the points $(0, 1)$ and $(\frac{1}{2(c-1)}, \frac{c}{2(c-1)})$. The line segment that connects the two is given by the equation $1 + q(2 - c)$ hence at $q = \frac{1}{2}$ it is $2 - c/2$, therefore $a_{D_2} = \frac{5}{4} - 2 + \frac{c}{2} = \frac{c}{2} - \frac{3}{4}$.

Setting $a_{D_1} = a_{D_2}$ yields $c = \frac{7}{4}$ with $a_{D_1} = a_{D_2} = \frac{1}{8}$. Therefore, for any auction that we decide to run will have additive loss of $2a = \frac{1}{4}$ on one of the two distributions, and we need $\Omega(2^n/n)$ samples to decide which distribution we are dealing with. □

**Corollary 2.19.** *For any $\epsilon > 0$, to obtain a $(\frac{7}{8} + \epsilon)$-approximation (multiplicative) to Myerson's auction, with constant probability you need $\Omega(2^n/n)$ samples.*

*Proof.* We can upper bound the revenue curve by the constant 2, to show that the optimal auction cannot have expected revenue more than 2. Since we have additive loss of $1/4$, the multiplicative approximation ratio is at most $\frac{7}{8}$. □

# Chapter 3

# Learning Utilities in Succinct Games

One of the central questions in game theory deals with predicting the behavior of agents, based on their utilities for different outcomes.[1] But in typical situations, we may be able to observe the behavior of agents, while their utilities are unobserved. This motivates looking at the inverse of the traditional problem: given the agents' equilibrium behavior, what are possible utilities that motivate this behavior? We consider this problem in arbitrary normal-form games in which the utilities can be represented by a small number of parameters, such as in graphical, congestion, and network design games. In all such settings, we show how to efficiently, i.e. in polynomial time, determine utilities consistent with a given correlated equilibrium. However, inferring both utilities and structural elements (e.g., the graph within a graphical game) is in general NP-hard. From a theoretical perspective our results show that rationalizing an equilibrium is computationally easier than computing it; from a practical perspective a practitioner can use our algorithms to validate behavioral models.

## 3.1 Introduction

One of the central and earliest questions in game theory deals with predicting the behavior of an agent. This question has led to the development of a wide range of theories and solution concepts —such as the Nash equilibrium— which determine the players' actions from their utilities. These predictions in turn may be used to inform economic analysis, improve artificial intelligence software, and construct theories of

[1]The work in this chapter is based on joint work with Volodymyr Kuleshov and was presented at WINE'15 [Kuleshov and Schrijvers, 2015]. Volodymyr Kuleshov was the first author on the paper.

human behavior.

Perhaps equally intriguing is the *inverse* of the above question: given data of the observed behavior of players in a game, how can we infer the utilities that led to this behavior? Surprisingly, this question has received much less attention, even though it arises just as naturally as its more famous converse.

For instance, inferring or *rationalizing* player utilities ought to be an important part of experimental protocols in the social sciences. An experimentalist should test the validity of their model by verifying whether it admits any utilities that are consistent with observed data. More ambitiously, the experimentalist may wish to develop predictive techniques, in which one tries to forecast the agents' behavior from earlier observations, with utilities serving as an intermediary in this process.

Inferring utilities also has numerous engineering applications. In economics, one could design mechanisms that adapt their rules after learning the utilities of their users, in order for instance to maximize profits. In machine learning, algorithms that infer utilities in a single-agent reinforcement learning setting are key tools for developing helicopter autopilots, and there exists ongoing research on related algorithms in the multi-agent setting.

### 3.1.1 Our Contributions.

Previous work on computational considerations for rationalizing equilibria has only considered specific settings, such as matching [Kalyanaraman and Umans, 2008] and network formation games [Kalyanaraman and Umans, 2009]. Here, we instead take a top-down approach and consider the problem in an *arbitrary* normal-form game. Although our results hold generally, the problem becomes especially interesting when the normal-form game is succinct, meaning that player utilities can be represented by a small number of parameters. The number of outcomes in an arbitrary game is in the worst case exponential in the number of players, so even storing the utilities would already require a prohibitive amount of storage. However, many games have additional structure which allows the utilities to be succinctly represented. Indeed, most games studied in the literature —including congestion, graphical, scheduling, and network design games— have this property. Within all succinct games, we establish two main results:

- When the structure of a game (e.g. the graph in a graphical game) is known, we can find utilities that rationalize the equilibrium using a small LP. The LP is polynomial rather than exponential in the number of players and their actions, and hence can be solved in polynomial time using the ellipsoid method. We discuss these results in Section 3.4.

- If the structure of a succinct game is unknown, inferring both utilities and the correct game structure is NP-hard. We discuss these results in Section 3.5.

### 3.1.2 Related Work

**Theoretical Computer Science.** Kalyanaraman et al. studied the computational complexity of rationalizing stable matchings [Kalyanaraman and Umans, 2008], and network formation [Kalyanaraman and Umans, 2009]. In the latter case, they showed that game attributes that are local to a player can be rationalized, while other, more global, attributes cannot; this mirrors our observations on the hardness of inferring utilities versus inferring game structure. The forward direction of our problem — computing an equilibrium from utilities — is a central question within algorithmic game theory. Computing Nash equilibria is intractable [Daskalakis et al., 2006] even for 2 player games [Chen et al., 2009] (and therefore may be a bad description of human behavior); correlated equilibria, however, are easy to compute in succinct games [Papadimitriou and Roughgarden, 2008] and can be found using simple iterative dynamics [Foster and Vohra, 1997, Hart and Mas-Colell, 2000]. Our results show that while a Nash equilibrium is hard to compute, it is easy to rationalize. For correlated equilibria, both computing and rationalizing it are feasible.

**Economics.** Literature on rationalizing agent behavior [Samuelson, 1948, Afriat, 1967, Varian, 1982] far predates computational concerns. The field of revealed preference [Varian, 2006] studies an agent who buys different bundles of a good over time, thus revealing more information about its utilities. These are characterized by sets of linear inequalities, which become progressively more restrictive; we adopt this way of characterizing agent utilities in our work as well, but in addition we prove that solving the problem can be done in polynomial time.

**Econometrics.** Recently, Nekipelov et al. [2015] discussed inferring utilities of bidders in online ad auctions, assuming bidders are using a no-regret algorithm for bidding. While no-regret learning agents do converge to a correlated equilibrium, the authors discuss a private-information game, rather than the full information games we consider.

The identification problem in econometrics includes formulations for games, e.g. [Bresnahan and Reiss, 1991, Lise, 2001, Bajari et al., 2010], but their goal is to find a single set of utilities that best describes observed behavior. Since this is often computationally infeasible, much of the literature proposes different estimators. From a theoretical perspective, for the class of succinct games we show that finding *a* set (not necessarily the most likely) of utilities is computationally feasible. Additionally

from a practical perspective, we uncover the entire space of valid utilities, which can be used to indicate how confident we should be about assumptions on a model.

**Inverse Reinforcement Learning.** Algorithms that infer the payoff function of an agent within a Markov decision process [Ng and Russell, 2000] are a key tool in building helicopter autopilots [Abbeel and Ng, 2004]. Our work establishes an analogous theory for multi-agent settings. Inverse reinforcement learning has also been used to successfully predict driver behavior in a city [Waugh et al., 2013, Ziebart et al., 2008]; but this work does not learn the utilities of players directly.

**Inverse Optimization.** Game theory can be interpreted as multi-player optimization, with different agents maximizing their individual objective functions. Recovering the objective function from a solution of a linear program can be solved using a *different* linear program [Ahuja and Orlin, 2001]. Our work considers the analogous inverse problem for multiple players and also solves it using an LP.

## 3.2 Preliminaries

In a normal-form game $G \triangleq [(A_i)_{i=1}^n, (\mathbf{u}_i)_{i=1}^n]$, a player $i \in \{1, 2, ..., n\}$ has $m_i$ actions $A_i \triangleq \{a_1^i, a_2^i, ..., a_{m_i}^i\}$ and utilities $\mathbf{u}_i \in \mathbb{R}^m$, where $m = \prod_{i=1}^n m_i$ is the cardinality of the joint-action space $A \triangleq \times_{i=1}^n A_i$. An $\mathbf{a} \in A$ is called a joint action of all the players and let $\mathbf{a}_{-i}$ be $\mathbf{a}$ with the action of player $i$ removed. A mixed strategy of player $i$ is a probability distribution $\mathbf{p}_i \in \mathbb{R}^{m_i}$ over the set of actions $A_i$. A correlated equilibrium (CE) of $G$ is a probability distribution $\mathbf{p} \in \mathbb{R}^m$ over $A$ that satisfies

$$\sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_j^i, \mathbf{a}_{-i}) \geq \sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_k^i, \mathbf{a}_{-i}) \tag{3.1}$$

for each player $i$ and each pair of actions $a_j^i, a_k^i$. This equation captures the idea that no player wants to unilaterally deviate from their equilibrium strategy. Correlated equilibria exist in every game, are easy to compute using a linear program, and arise naturally from the repeated play of learning players [Foster and Vohra, 1997, Hart and Mas-Colell, 2000].

A (mixed) Nash equilibrium is a correlated equilibrium $\mathbf{p}$ that is a product distribution $p(\mathbf{a}) = p_1(a_1) \times ... \times p_n(a_n)$, where the $\mathbf{p}_i \in \mathbb{R}^{m_i}$ are mixed player strategies. In a Nash equilibrium, each player chooses their own strategy (hence the product form), while in a correlated equilibrium the players' actions can be viewed as coming from an outside mediator. A Nash equilibrium exists in every game, but is hard to compute even in the 2-player setting [Chen et al., 2009].

## 3.3 Succinct Games

In general, the dimension $m$ of player $i$'s utility $\mathbf{u}_i$ is exponential in the number of players: if each player has $t$ actions, $\mathbf{u}_i$ specifies a value for each of their $t^n$ possible combinations. Therefore, we restrict our attention to games $G$ that have a special structure which allows the $\mathbf{u}_i$ to be parametrized by a small number of parameters $\mathbf{v}$; such games are called *succinct* [Papadimitriou and Roughgarden, 2008].

A classical example of a succinct game is a *graphical game*, in which there is a graph $H$ with a node for every player, and the utility of a player depends only on itself and the players on incident nodes in $H$. Let $k$ be the number of neighbors of $i$ in $H$, then we only need to specify the utility of $i$ for each combination of actions of $k+1$ players (rather than $n$). For bounded-degree graphs this greatly reduces the number of parameter value. If the maximum degree in the graph is $k$ and each player has at most $t$ actions, then the total number of utility values per player is at most $t^{k+1}$, which is independent of $n$.

**Definition 3.1.** *A succinct game*

$$G \triangleq [(A_i)_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (F_i)_{i=1}^n]$$

*is a tuple of sets of player actions $A_i$, parameters $\mathbf{v}_i \in \mathbb{R}^d$, and functions $F_i : \mathbb{R}^d \times A \to \mathbb{R}$ that compute the utility $u_i(\mathbf{a}) = F_i(\mathbf{v}_i, \mathbf{a})$ of a joint action $\mathbf{a}$.*

We will further restrict our attention to succinct games in which the $F_i$ have a particular linear form. As we will soon show, almost every succinct game in the literature is also linear. This definition will in turn enable a simple and unified mathematical analysis across all succinct games.

**Definition 3.2.** *A linear succinct game*

$$G \triangleq [(A_i)_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (O_i)_{i=1}^n]$$

*is a succinct game in which the utilities $\mathbf{u}_i$ are specified by $\mathbf{u}_i = O_i\mathbf{v}_i$, where $O_i \in \{0,1\}^{m \times d}$ is an* outcome matrix *mapping parameters into utilities.*

Note that a linear succinct game is a special case of Definition 3.1 with $F_i(\mathbf{v}_i, \mathbf{a}) = (O_i\mathbf{v}_i)_{\mathbf{a}}$, which is the component of $O_i\mathbf{v}$ corresponding to $\mathbf{a}$.

The outcome matrix $O_i$ has an intuitive interpretation. We can think of a set of $d$ distinct outcomes $\mathcal{O}_i$ that can affect the utility of player $i$. The parameters $\mathbf{v}_i$ specify a utility $\mathbf{v}_i(o)$ for each outcome $o \in \mathcal{O}_i$. When a joint action $\mathbf{a}$ occurs, it results in the realization of a subset $\mathcal{O}_i(\mathbf{a}) \triangleq \{o : (O_i)_{\mathbf{a},o} = 1\}$ of the outcomes, specified by the positions of the non-zero entries of matrix $O_i$. The utility $u_i(\mathbf{a}) = (O_i\mathbf{v}_i)_{\mathbf{a}}$ equals the

sum of valuations of the realized outcomes:

$$u_i(\mathbf{a}) = \sum_{o \in \mathcal{O}_i(\mathbf{a})} v_i(o).$$

Graphical games, which we discussed above, are an example of a succinct game that is linear. In a graphical game with an associated graph $H$, outcomes correspond to joint actions $\mathbf{a}_{N(i)} = (a^{(k)})_{k \in N(i)}$ by $i$ and its neighbors in $H$. A joint-action $\mathbf{a}$ activates the single outcome $o$ that is associated to a $\mathbf{a}_{N(i)}$ in which the actions are specified by $\mathbf{a}$. The matrix $O_i$ is defined as

$$(O_i)_{\mathbf{a},\mathbf{a}_{N(i)}} = \begin{cases} 1 & \text{if } \mathbf{a}, \mathbf{a}_{N(i)} \text{ agree on the actions of } N(i) \\ 0 & \text{otherwise.} \end{cases}$$

### 3.3.1 Succinct Representations of Equilibria

Since there is an exponential number of joint actions, a correlated equilibrium $\mathbf{p}$ (which is a distribution over joint actions) may require exponential space to write down. To make sure that the input is polynomial in $n$, we require that $\mathbf{p}$ be represented as a polynomial mixture of product distributions (PMP) $\mathbf{p} = \sum_{k=1}^{K} \mathbf{q}_k$, where $K$ is polynomial in $n$, $q_k(\mathbf{a}) = \prod_{i=1}^{n} q_{ik}(a_i)$ and $q_{ik}$ is a distribution over $A_i$. Correlated equilibria in the form of a PMP exist in every game and can be computed efficiently [Papadimitriou and Roughgarden, 2008]. A Nash equilibrium is already a product distribution, so it is a PMP with $K = 1$.

### 3.3.2 What it Means to Rationalize Equilibria

Finding utilities consistent with an equilibrium $\mathbf{p}$ amounts to finding $\mathbf{u}_i$ that satisfy Equation 3.1 for each player $i$ and for each pair of actions $a_j^i, a_k^i \in A_i$. It is not hard to show that Equation 3.1 can be written in matrix form as

$$\mathbf{p}^T C_{ijk} \mathbf{u}_i \geq 0, \tag{3.2}$$

where $C_{ijk}$ is an $m \times m$ matrix that has the form

$$(C_{ijk})_{(\mathbf{a}_{\text{row}},\mathbf{a}_{\text{col}})} = \begin{cases} -1 & \text{if } \mathbf{a}_{\text{row}} = (a_j, \mathbf{a}_{-i}^{\text{col}}) \\ 1 & \text{if } \mathbf{a}_{\text{row}} = (a_k, \mathbf{a}_{-i}^{\text{col}}) \\ 0 & \text{otherwise.} \end{cases}$$

This formulation exposes intriguing symmetry between the equilibrium distribution $\mathbf{p}$ and the utilities $\mathbf{u}_i$. By our earlier definitions, the utilities $\mathbf{u}_i$ in a linear succinct

game can be written as $\mathbf{u}_i = O_i \mathbf{v}_i$; this allows us to rewrite Equation 3.2 as

$$\mathbf{p}^T C_{ijk} O_i \mathbf{v}_i \geq 0. \tag{3.3}$$

While $C_{ijk}$ and $O_i$ are exponentially large in $n$, their product is not, so in Section 3.4 we show that we can compute this product efficiently, without constructing $C_{ijk}$ and $O_i$ explicitly.

### 3.3.3 Non-Degeneracy Conditions

In general, inferring agent utilities is not a well-defined problem. For instance, Equation 3.1 is always satisfied by $\mathbf{v}_i = \mathbf{0}$ and remains invariant under scalar multiplication $\alpha \mathbf{v}_i$. To avoid such trivial solutions, we add an additional non-degeneracy condition on the utilities.

**Condition 1** (Non-degeneracy)**.** *A non-degenerate vector* $\mathbf{v} \in \mathbb{R}^d$ *satisfies* $\sum_{k=1}^{d} v_k = 1$.

### 3.3.4 The Inverse Game Theory Problem

We are now ready to formalize two important inverse game theory problems. In the first problem — Inverse-Utility — we observe $L$ games between $n$ players; the structure of every game is known, but can vary. As a motivating example, consider $n$ drivers that play a congestion game each day over a network of roads and on certain days some roads may be closed. Or consider $L$ scheduling games where different subsets of machines are available on each day. Our goal is to find valuations that rationalize the observed equilibrium of each game.

**Definition 3.3** (Inverse-Utility problem)**.** *Given:*

1. *A set of $L$ partially observed succinct $n$-player games $G_l = [(A_{il})_{i=1}^n, \cdot, (O_{il})_{i=1}^n]$, for $l \in \{1, 2, ..., L\}$.*
2. *A set of $L$ correlated equilibria $(\mathbf{p}_l)_{l=1}^L$.*

*Determine succinct utilities $(\mathbf{v}_i)_{i=1}^n$ such that $\mathbf{p}_l$ is a valid correlated equilibrium in each $G_l$, in the sense that Equation 3.3 holds for all $i$ and for all $a_j^i, a_k^i \in A_{il}$. Alternatively, report that no such $\mathbf{v}_i$ exist.*

In the second problem — Inverse-Game — the players are again playing in $L$ games, but this time both the utilities and the structure of these games are unknown.

**Definition 3.4** (Inverse-game problem)**.** *Given:*

1. *A set of $L$ partially observed succinct $n$-player games $G_l = [(A_{il})_{i=1}^n, \cdot, \cdot]$, for $l \in \{1, 2, ..., L\}$.*
2. *A set of $L$ correlated equilibria $(\mathbf{p}_l)_{l=1}^m$.*
3. *Candidate game structures $(\mathcal{S}_l)_{l=1}^L$, one $\mathcal{S}_l$ per game. Each $\mathcal{S}_l = (S_{lh})_{h=1}^p$ contains $p$ candidate structures. A structure $S_{lh} = (O_{lhi})_{i=1}^n$ specifies an outcome matrix $O_{lhi}$ for each player $i$.*

*Determine succinct utilities $(\mathbf{v}_i)_{i=1}^n$ and a structure $S_l^* = (O_{li}^*)_{i=1}^n \in \mathcal{S}_l$ for each game, such that $\mathbf{p}_l$ is a correlated equilibrium in each $[(A_{il})_{i=1}^n, (\mathbf{v}_i)_{i=1}^n, (O_{li}^*)_{i=1}^n]$, in the sense that*

$$\mathbf{p}_l^T C_{ijk} O_{il}^* \mathbf{v}_i \geq 0$$

*holds for all $i, l$ and for all $a_j^i, a_k^i \in A_{il}$. Alternatively, report that no such $\mathbf{v}_i$ exist.*

An example of this problem is when we observe $L$ graphical games among $n$ players and each game has a different and unknown underlying graph chosen among a set of candidates. We wish to infer both the common $\mathbf{v}$ and the graph of each game.

## 3.4 Learning Utilities in Succinct Games

In this section, we show how to solve Inverse-Utility in most succinct games. We start by looking at a general linear succinct game, and derive a simple condition under which Inverse-Utility can be solved. Then we consider specific cases of games (e.g. graphical, congestion, network games), and show 1) that they are succinct and linear, and 2) that they satisfy the previous condition.

### 3.4.1 General Linear Succinct Games

To solve Inverse-Utility, we need to find valuations $\mathbf{v}_i$ that satisfy the equilibrium condition (3.3) for every player $i$ and every pair of actions $a_j^i, a_k^i$. Notice that if we can compute the product $\mathbf{c}_{ijk}^T \triangleq \mathbf{p}^T C_{ijk} O_i$, then Equation 3.3 reduces to a simple linear constraint $\mathbf{c}_{ijk}^T \mathbf{v}_i \leq 0$ for $\mathbf{v}_i$. However, the dimensions of $C_{ijk}$ and $O_i$ grow exponentially with $n$; in order to multiply these objects we must therefore exploit special problem structure. This structure exists in every game for which the following simple condition holds.

**Property 3.5.** *Let $A_i(o) = \{\mathbf{a} : (O_i)_{\mathbf{a},o} = 1\}$ be the set of joint-actions that trigger outcome $o$ for player $i$. The equilibrium summation property holds if*

$$\sum_{\mathbf{a}_{-i}:(a_j^i, \mathbf{a}_{-i}) \in A_i(o)} p(\mathbf{a}_{-i}) \tag{3.4}$$

*can be computed in polynomial time for any outcome $o$, product distribution $\mathbf{p}$, and action $a_j^i$.*[2]

**Lemma 3.6.** *Let $G$ be a linear succinct game and let $\mathbf{p}$ be a PMP correlated equilibrium. Let $\mathbf{c}_{ijk}^T \triangleq \mathbf{p}^T C_{ijk} O_i$ be the constraint on vector $\mathbf{v}_i$ in Equation 3.3 for a pair of actions $a_k^i, a_j^i$. If Property 3.5 holds, then the components of $\mathbf{c}_{ijkj}^T$ can be computed in polynomial time.*

*Proof.* For greater clarity, we start with the formulation (3.1) of constraint (3.3):

$$\sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_j^i, \mathbf{a}_{-i}) \geq \sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u(a_k^i, \mathbf{a}_{-i}) \tag{3.5}$$

We derive from (3.5) an expression for each component of $\mathbf{c}_{ijk}$.

Recall that we associate the components of $\mathbf{v}_i$ with a set of outcomes $\mathcal{O}_i$. Let $\mathcal{O}_i(\mathbf{a}) = \{o : O_{(\mathbf{a},o\}} = 1\}$ denote the set of outcomes that are triggered by $\mathbf{a}$; similarly, let $A(o) = \{\mathbf{a} : (O_i)_{\mathbf{a},o} = 1\}$ be the set of joint-actions that trigger an outcome $o$. The left-hand side of (3.5) can be rewritten as:

$$\begin{aligned}
\sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u_i(a_j^i, \mathbf{a}_{-i}) &= \sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) \sum_{o \in \mathcal{O}_i(a_j^i, \mathbf{a}_{-i})} v_i(o) \\
&= \sum_{o \in \mathcal{O}_i} \sum_{\substack{\mathbf{a}_{-i}: \\ (a_j^i, \mathbf{a}_{-i}) \in A_i(o)}} p(a_j^i, \mathbf{a}_{-i}) v_i(o) \\
&= \sum_{o \in \mathcal{O}_i} v_i(o) \sum_{\substack{\mathbf{a}_{-i}: \\ (a_j^i, \mathbf{a}_{-i}) \in A_i(o)}} p(a_j^i, \mathbf{a}_{-i})
\end{aligned}$$

Similarly, the right-hand side of (3.5) can be rewritten as

$$\sum_{\mathbf{a}_{-i}} p(a_j^i, \mathbf{a}_{-i}) u_i(a_k^i, \mathbf{a}_{-i}) = \sum_{o \in \mathcal{O}_i} v_i(o) \sum_{\substack{\mathbf{a}_{-i}: \\ (a_k^i, \mathbf{a}_{-i}) \in A_i(o)}} p(a_j^i, \mathbf{a}_{-i}).$$

Substituting these two expressions into (3.5) and factoring out $p_i(a_j^i)$ (recall that $\mathbf{p}$

[2]Property 3.5 is closely related to the *polynomial expectation property* (PEP) of [Papadimitriou and Roughgarden, 2008] which states that the expected utility of a player in a succinct game should be efficiently computable for a product distribution. In fact, the arguments we will use to show that this property holds are inspired by arguments for establishing the PEP.

is a product distribution) allows us to rewrite (3.5) as:

$$\sum_{o \in \mathcal{O}_i} v_i(o) \left[ \sum_{\substack{\mathbf{a}_{-i}: \\ (a_j^i, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) - \sum_{\substack{\mathbf{a}_{-i}: \\ (a_k^i, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) \right] \geq 0.$$

Notice that the expression in brackets corresponds to the entries of the vector $\mathbf{c}_{ijk}^T$. If $\mathbf{p}$ is a product distribution, then by Property 3.5, we can compute these terms in polynomial time. If $\mathbf{p}$ is a correlated equilibrium with a PMP representation $\sum_{k=1}^{K} q_k$, it is easy to see that by linearity of summation we can apply Property 3.5 $K$ times on each of the terms $q_k$ and sum the results. This establishes the lemma. □

Lemma 3.6 suggests solving INVERSE-UTILITY in a game $G$ by means of the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} f(\mathbf{v}_i) && && (3.6) \\ \text{subject to} \quad & \mathbf{c}_{ijk}^T \mathbf{v}_i \geq 0 && \forall i, j, k && (3.7) \\ & \mathbf{1}^T \mathbf{v}_i = 1 && \forall i && (3.8) \end{aligned}$$

Constraint (3.7) ensures that $\mathbf{p}$ is a valid equilibrium; by Lemma 3.6, we can compute the components of $c_{ijk}$ if Property 3.5 holds in $G$. Constaint (3.8) ensures that the $\mathbf{v}_i$ are non-degenerate. The objective function (3.6) selects a set of $\mathbf{v}_i$ out of the polytope of all valid utilities. It is possible to incorporate into this program additional prior knowledge on the form of the utilities or on the coupling of valuations across players.

The objective function $f$ may also incorporate prior knowledge, or it can serve as a regularizer. For instance, we may choose $f(\mathbf{v}_i) = ||\mathbf{v}_i||_1$ to encourage sparsity and make the $\mathbf{v}_i$ more interpretable. We may also use $f$ to avoid degenerate $\mathbf{v}_i$'s; for instance, in graphical games, $\mathbf{c}_{ijk}^T \mathbf{1} = \mathbf{0}$ and constant $\mathbf{v}_i$'s are a valid solution. We may avoid this by adding the $\mathbf{v} \geq 0$ constraint (this is w.l.o.g. when $\mathbf{c}_{ijk}^T \mathbf{1} = \mathbf{0}$) and by choosing $f(\mathbf{v}) = \sum_{o \in \mathcal{O}_i} v(o) \log v(o)$ to maximize entropy.

Note that to simply find a valid $\mathbf{v}_i$, we may set $f(\mathbf{v}_i) = 0$ and find a feasible point via linear programming. Moreover, if we observe $L$ games, we simply combine the constraints $\mathbf{c}_{ijk}$ into one program. Formally, this establishes the main lemma of this section:

**Lemma 3.7.** *The* INVERSE-GAME *problem can be solved efficiently in any game where Property 3.5 holds.*

### 3.4.2 Inferring Utilities in Popular Succinct Sames

We now turn our attention to specific families of succinct games which represent the majority of succinct games in the literature [Papadimitriou and Roughgarden, 2008]. We show that these games are linear and satisfy Property 3.5; so that, INVERSE-UTILITY can be solved using the optimization problem (3.6).

**Graphical Games.** In graphical games [Kearns et al., 2001], a graph $H$ is defined over the set of players; the utility of a player depends only on their actions and those of its neighbors in the graph.

The outcomes for player $i$ are associated to joint-actions $a_{N(i)}$ by the set containing $i$ and its neighbors $N(i)$. A joint-action $\mathbf{a}$ triggers the outcome $a_{N(i)}$ specified by actions of the players in $N(i)$ in $\mathbf{a}$. Formally,

$$(O_i)_{\mathbf{a},\mathbf{a}_{N(i)}} = \begin{cases} 1 & \text{if } \mathbf{a}, \mathbf{a}_{N(i)} \text{ agree on the actions of } N(i) \\ 0 & \text{otherwise.} \end{cases}$$

It is easy to verify that graphical games possess Property 3.5. Indeed, for any outcome $o = \mathbf{a}_{N(i)}$ and action $a^i_j$, and letting $a^k_{N(i)}$ be the action of player $k$ in $\mathbf{a}_{N(i)}$, we have

$$\begin{aligned} \sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) &= \prod_{\substack{k \in N(i) \\ k \neq i}} p_k(a^k_{N(i)}) \prod_{\substack{k \notin N(i) \\ k \neq i}} \sum_{a^k \in A_k} p_k(a^k) \\ &= \prod_{\substack{k \in N(i) \\ k \neq i}} p_k(a^k_{N(i)}) \end{aligned}$$

**Polymatrix Games.** In a polymatrix game [Howson Jr, 1972], each player plays $i$ in $(n-1)$ simultaneous 2-player games against each of the other players, and utilities are summed across all these games. Formally, each joint-action triggers $n-1$ different outcomes for player $i$, one for each pair of actions $(a^i, a^j)$ and thus $u_i(\mathbf{a}) = \sum_{j \neq i} v_i(a^i, a^j)$. The associated outcome matrix is

$$(O_i)_{\mathbf{a},(a^i,a^j)} = \begin{cases} 1 & \text{if } a^i_j \text{ and } a^i_i \text{ are played within } \mathbf{a} \\ 0 & \text{otherwise.} \end{cases}$$

To establish Property 3.5, observe that when $o = (a^i, a^j)$ is one of the outcomes affecting the utility of player $i$, we have

$$\sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) = \sum_{\mathbf{a}_{-i}: a^j \in \mathbf{a}_{-i}} p(\mathbf{a}_{-i}) = p_j(a^j).$$

**Hypergraphical Games.** Hypergraphical games [Papadimitriou and Roughgarden, 2008] generalize polymatrix games to the case where the simultaneous games involve potentially more than two players. Each instance of a hypergraphical game is associated with a hypergraph $H$; the vertices of $H$ correspond to players and a hyperedge $e$ indicates that the players connected by $e$ play together in a subgame; the utility of player $i$ is the sum its utilities in all the subgames in which it participates.

The fact that hypergraphical games are linear and possess Property 3.5 follows easily from our discussion of polymatrix and graphical games.

**Congestion Games.** In congestion games [Rosenthal, 1973], players compete for a set of resources $E$ (e.g., roads in a city, modeled by edges in a graph); the players' actions correspond to subsets $a^i \subseteq E$ of the resources. After all actions have been played, each player $i$ incurs a cost that equals the sum $\sum_{e \in a^i} d_e(l_e)$ of delays $d_e(\ell_e)$ at each resource $e$, where $\ell_e(\mathbf{a}) = |\{i : e \in a^i\}|$ denotes the number of players using that resource. In the example involving roads, delays indicate how long it takes to traverse a road based on the congestion.

The outcomes for player $i$ in congestion games are associated with a resource $e$ and the number $L$ of players using that resource; we denote this by $o = (e, L)$. A joint action $\mathbf{a}$ activates the outcomes for the resources in $a^i$ that have $\ell_e(\mathbf{a})$ users. The value $v(o)$ of an outcome $o = (e, L)$ corresponds to the delay experienced on $e$. Formally, the outcome matrix for a congestion game has the form

$$(O_i)_{\mathbf{a},(e,L)} = \begin{cases} 1 & \text{if } e \in a^i \text{ and } \ell_e(\mathbf{a}) = L \\ 0 & \text{otherwise.} \end{cases}$$

To establish Property 3.5, we need to show that the expression

$$\sum_{\substack{\mathbf{a}_{-i}: \\ (a^i_j, \mathbf{a}_{-i}) \in A_i(o)}} p(\mathbf{a}_{-i}) = \sum_{\mathbf{a}_{-i}: \ell(\mathbf{a}_{-i}) = L - 1\{e \in a^i_j\}} p(\mathbf{a}_{-i})$$

can be computed for any outcome $o = (e, L)$. Here, $\ell(\mathbf{a}_{-i})$ denotes the number of players other than $i$ using resource $e$ and $1\{e \in a^i_j\}$ equals one if $e \in a^i_j$ and zero

otherwise.

The expression $P_L(e) \triangleq \sum_{\mathbf{a}_{-i}:\ell(\mathbf{a}_{-i})=L} p(\mathbf{a}_{-i})$ can be computed via dynamic programming. Indeed, observe that $P_L(e)$ equals $P[\sum_{j\neq i} B_j(p,e) = L]$, where $B_j(p,e)$ is a Bernoulli random variable whose probability of being one corresponds to the probability $P_{j,e} \triangleq \sum_{a^j:e\in a^j} p_j(a^j)$ of player $j$ selecting an action that includes $e$. The probabilities $P_{j,e}$ are of course easy to compute. From the $P_{j,e}$ it is easy to compute the $P_L(e)$ using dynamic programming via the recursion:

$$P_L(e) = \sum_{j\neq i} P\left[B_j(p,e) = 1 \cap B_k(p,e) = 0\ \forall k \neq i,j\right] P_{L-1}(e).$$

**Facility Location and Network Design Games.** In facility location games [Chun et al., 2004], players choose one of multiple facility locations, each with a certain cost, and the cost of each facility is then divided by all the players who build it. In network design games [Anshelevich et al., 2008], players choose paths in a graph to connect their terminals, and the cost of each edge is shared among the players that use it.

These two game types are special cases of congestion games with particular delay functions. These can be handled through additional linear constraints. The earlier discussion for congestion games extends easily to this setting to establish Property 3.5.

**Scheduling Games.** In a scheduling game [Fotakis et al., 2002, Papadimitriou and Roughgarden, 2008], there are $M$ machines and each player $i$ schedules a job on a machine $a^i$; the job has a machine-dependent running time $t(m,i)$. The player then incurs a cost $t_i(\mathbf{a}) = \sum_{\{j:a^j=a^i\}} t(a^i,j)$ that equals the sum of the running times of all tasks on its machine.

Player outcomes $o = (m,j)$ are associated with a machine $m$ and the task of a player $j$. The outcome matrix $O_i$ has the form

$$(O_i)_{\mathbf{a},(m,j)} = \begin{cases} 1 & \text{if } m \in a^i \text{ and } m \in a^j \\ 0 & \text{otherwise.} \end{cases}$$

Property 3.5 can be established by adapting the dynamic programing argument used for congestion games. Note also that congestion games require adding the constraint $v_i(m,k) = v_j(m,k)$ for all $i$ and $j$ in optimization problem (3.6). We summarize our results in the following theorem.

**Theorem 3.8.** *The* Inverse-Utility *problem can be solved in polynomial time for*

*the classes of succinct games defined above.*

## 3.5 Learning the Structure of Succinct Games

Next, we consider the INVERSE-GAME problem. Unlike in the previous section which contained sweeping positive results, here we give a natural setting in which INVERSE-GAME is hard to solve, while the corresponding instance of INVERSE-UTILITY is easy.

We establish this result using the following additional non-degeneracy condition on the player utilities.

**Condition 2** (Non-indifference)**.** *Parameters* $\mathbf{v}_i$ *satisfy the non-indifference condition if there exist* $a_j^i, a_k^i, \mathbf{a}_{-i}$ *such that* $u_i(a_j^i, \mathbf{a}_{-i}) \neq u_i(a_k^i, \mathbf{a}_{-i})$*, where* $\mathbf{u}_i = O_i \mathbf{v}_i$

**Theorem 3.9.** *Assuming Condition 2, it is NP-Hard to solve* INVERSE-GAME *in the setting of graphical games. However, the corresponding instance of* INVERSE-UTILITY *is easy to solve.*

*Proof.* We reduce from an instance of 3-SAT in which for every variable appears in at most $m-2$ clauses.[3] Given an instance of 3-SAT with $m$ clauses and $n$ variables, we construct an instance of INVERSE-GAME as follows.

There are $n+1$ players in each game $j$ (for $1 \leq j \leq m$) that are indexed by $i = 0, .., n$. Player 0 has only one action: $a^{(0)}$. Every other player $i \geq 1$ has 2 actions: $a_T^{(i)}$ and $a_F^{(i)}$.

Every game $j$ is associated with a clause $C_j$. Game $j$ has an unknown underlying graph that is chosen in the set of graphs $\mathcal{S}_j = \{H_{j1}, H_{j2}, H_{j3}\}$, where $H_{jk}$ is the graph consisting of only a single edge between player 0 and the player associated with the variable that appears as the $k$-th literal in clause $j$. In other words, in each game, only one of three possible players is connected to player 0 by an edge.

The utilities $v_i$ of each player $i \geq 1$ are four-dimensional: they specify two values $v_i(a_T^{(i)}), v_i(a_F^{(i)})$ when player $i$ is not connected by an edge to player 0, and two values $v_i(a_T^{(i)}; a^{(0)}), v_i(a_F^{(i)}; a^{(0)})$ when they are.

For every clause $C_j$, we also define an input equilibrium $p_j$. Each $p_j$ is a pure strategy Nash equilibrium and decomposes into a product $p_j = \prod_{i=1}^n p_{ji}$. Since player 0 has only one action, $p_{j0}$ is defined trivially. When variable $x_i$ appears in clause $C_j$, we define the probability of player $i \geq 1$ playing action $a_T^{(i)}$ as

$$p_{ji}\left(a^{(i)} = a_T^{(i)}\right) = \begin{cases} 1 & \text{if } x_i \text{ is positively in clause } C_j \\ 0 & \text{if } x_i \text{ is negated in clause } C_j, \end{cases}$$

[3]This is without loss of generality: if there is a variable that appears in all clauses, we can construct two 2-SAT instances which can be solved efficiently, if a variable appears in all but one clause we can duplicate that clause.

and $p_{ji}(a^{(i)} = a_F^{(i)}) = 1 - p_{ji}(a^{(i)} = a_T^{(i)})$. When variable $x_i$ does not appear in clause $C_j$, we set the strategy in one such game $j$ (chosen arbitrarily) to be $p_{ji}(a^{(i)} = a_T^{(i)}) = 1$, and in the remaining games we set $p_{ji}(a^{(i)} = a_F^{(i)}) = 1$.

This completes the construction of the game. Next, we will introduce some notation and make a few observations, before showing that 3-SAT is encoded in the constructed game.

Let $\Delta_i(a_T^{(i)} \to a_F^{(i)}) \triangleq v_i(a_F^{(i)}, a^{(0)}) - v_i(a_T^{(i)}, a^{(0)})$ be the gain to player $i \geq 1$ for deviating from $a_T^{(i)}$ to $a_F^{(i)}$ when they are not connected to player 0, and let $\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) \triangleq v_i(a_F^{(i)}, a^{(0)}; a^{(0)}) - v_i(a_T^{(i)}, a^{(0)}; a^{(0)})$ be the gain when they are.

Observe that the constraint of player $i \geq 1$ when they are not connected to player 0 is of the form

$$p_{ij}(a_F^{(i)})\Delta_i(a_T^{(i)} \to a_F^{(i)}) + p_{ij}(a_T^{(i)})\Delta_i(a_F^{(i)} \to a_T^{(i)}) \geq 0,$$

and the constraint when $i$ and 0 are connected is similar.

Because each variable $x_i$ appear in at most $m-2$ clauses, and because of how we defined the probabilities, the following constraints act on $\Delta_i(a_T^{(i)} \to a_F^{(i)})$: in one game $j_1$ such that $x_i$ is not in $C_j$ we have $\Delta_i(a_T^{(i)} \to a_F^{(i)}) \geq 0$, and in all other such games we have $\Delta_i(a_T^{(i)} \to a_F^{(i)}) \leq 0$. Because by definition $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = -\Delta_i(a_F^{(i)} \to a_T^{(i)})$, we must have $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = \Delta_i(a_F^{(i)} \to a_T^{(i)}) = 0$. Because of non-degeneracy constraints on the utilities $v$, this implies that $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) \neq 0$. This concludes the observations.

We now show how 3-SAT is encoded in the game we defined. Suppose that we have an valid assignment of utilities and a structure; this leads to a satisfying assignment in 3-SAT: we simply set to be true in clause $j$ the literal associated with the player that is connected to player 0 in game $j$.

Clearly there will be one true literal per clause. We only need to show that both the literal and its negation are never chosen. Suppose that was the case and there were two clauses $j_1, j_2$ such that $x_i$ is chosen in $j_1$ and $\bar{x}_i$ is chosen in $j_2$. Then, player $i$'s constraint in $j_1$ is $\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$ and in $j_2$ it is $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) = -\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$ and there cannot be utilities that satisfy both these constraints.

Finally we show that a satisfying 3-SAT assignment leads to valid utilities in INVERSE-GAME. First, set all utilities $v_i(a_T^{(i)}), v_i(a_F^{(i)})$ to zero. Set the utility of player 0 to one. Pick a true literal in each clause $j$ and connect the corresponding player in game $j$ to player 0. We have to show that we can find valid utilities for all players. Clearly, that is feasible for player 0. We claim that it is also possible for any player $i \geq 1$. First, set $\Delta_i(a_T^{(i)} \to a_F^{(i)}) = -\Delta_i(a_F^{(i)} \to a_T^{(i)}) = 0$. Next, notice if $x_i$ is true, then all constraints involving player $i$ are $\Delta_i(a_T^{(i)} \to a_F^{(i)}; a^{(0)}) > 0$), and if $x_i$ is false,

all constraints involving player $i$ are $\Delta_i(a_F^{(i)} \to a_T^{(i)}; a^{(0)}) > 0$. In both cases we can find valid utilities to satisfy these constraints, and so INVERSE-GAME can be solved.

Finally, observe that the corresponding instance of INVERSE-UTILITY (i.e., the one in which the correct graph is pre-specified in advance) is easy to solve. The number of players and actions is very small. Furthermore, it is easy to enforce Condition 2, as there are only two actions for each player, and two values of $a_{-i}^i$ to consider; the condition can thus be enforced by adding a small (polynomial) number of constraints to the problem. □

# Chapter 4

# Anomaly Detection in a Stream

So far we have looked at situations in which more data leads to better results.[1] However, in practical applications, if we observe more data, the probability of including incorrect data also grows. For example, there could have been a mechanical failure during writing, or the data could have been corrupted after it was stored. This is not just true in game-theoretical applications, such as the ones described in the previous two chapters, but more generally in any application that uses data from the real world as input. Therefore, we phrase our results in this chapter quite generally, without restriction to data that came from game-theoretical applications.

We focus on the anomaly detection problem for dynamic data streams. While many algorithms have been proposed for this fundamental problem, there has been significantly less progress in anomaly detection in the non-parametric setting, i.e, where we do not have parametric models of the underlying distribution. As a related consequence performance of the most promising algorithms have not always been studied rigorously, and therefore these algorithms can fail in fairly innocuous instances. We view anomaly detection through the lens of random cut forests. We investigate a robust random cut data structure that can be used as a sketch or synopsis of the input stream. We provide a plausible definition of non-parametric anomalies based on the influence of an unseen point on the remainder of the data, i.e., the externality imposed by that point. We show how the sketch can be efficiently updated in a dynamic data stream. We demonstrate the viability of the algorithm on publicly available real data.

[1] This chapter is based on joint work with Nina Mishra, Sudipto Guha and Roy Gourav and was presented at ICML'16 [Guha et al., 2016]. It is currently implemented in Amazon Web Services.

## 4.1 Introduction

Anomaly detection is one of the cornerstone problems in data mining. Even though the problem has been well studied over the last few decades, the emerging explosion of data from online behavior and sensors connected to the internet of things leads us to reconsider the problem. In most of these contexts the data is streaming and well-understood prior models do not exist. Furthermore the input streams need not be append only, there may be corrections, updates and a variety of other dynamic changes. Two central questions in this regard are (1) how do we define anomalies? and (2) what data structure do we use to efficiently detect anomalies over dynamic data streams? In this chapter we initiate the formal study of both of these questions. For (1), we view the problem from the perspective of model complexity and say that a point is an anomaly if the complexity of the model increases substantially with the inclusion of the point. The labeling of a point is data dependent and corresponds to the externality imposed by the point in explaining the remainder of the data. We extend this notion of externality to handle "outlier masking" that often arises from duplicates and near duplicate records. Note that the notion of model complexity has to be amenable to efficient computation in dynamic data streams. This relates question (1) to question (2) which we discuss in greater detail next. However it is worth noting that anomaly detection is not well understood even in the simpler context of static batch processing and (2) remains relevant in the batch setting as well.

For question (2), we explore a randomized approach, akin to [Liu et al., 2012], due in part to the practical success reported in [Emmott et al., 2013]. Randomization is a powerful tool and known to be valuable in supervised learning [Breiman, 2001]. But its technical exploration in the context of anomaly detection is not well-understood and the same comment applies to the algorithm put forth in [Liu et al., 2012]. Moreover that algorithm has several limitations as described in Section 4.4.1. In particular, we show that in the presence of irrelevant dimensions, crucial anomalies are missed. In addition, it is unclear how to extend this work to a stream. Prior work attempted solutions [Tan et al., 2011] that extend to streaming, however those were not found to be effective [Emmott et al., 2013]. To address these limitations, we put forward a sketch or synopsis termed *robust random cut forest* (RRCF) formally defined as follows.

**Definition 4.1.** *A* **robust random cut tree** *(RRCT) on point set $S$ is generated as follows:*

1. *Choose a random dimension proportional to $\frac{\ell_i}{\sum_j \ell_j}$, where $\ell_i = \max_{x \in S} x_i - min_{x \in S} x_i$.*
2. *Choose $X_i \sim Uniform[\min_{x \in S} x_i,\ \max_{x \in S} x_i]$*

*3. Let $S_1 = \{x | x \in S, x_i \leq X_i\}$ and $S_2 = S \setminus S_1$ and recurse on $S_1$ and $S_2$.*

*A robust random cut forest (RRCF) is a collection of independent RRCTs.*

The approach in [Liu et al., 2012] differs from the above procedure in Step (1) and chooses the dimension to cut uniformly at random. We discuss this algorithm in more detail in Section 4.4.1 and provide extensive comparison.

Following question (2), we ask: Does the RRCF data structure contain sufficient information that is independent of the specifics of the tree construction algorithm? In this paper we prove that the RRCF data structure approximately preserves distances in the following sense:

**Theorem 4.2.** *Consider the algorithm in Definition 4.1. Let the weight of a node in a tree be the corresponding sum of dimensions $\sum_i \ell_i$. Given two points $u, v \in S$, define the tree distance between $u$ and $v$ to be the weight of the least common ancestor of $u, v$. Then the tree distance is always at least the Manhattan distance $L_1(u, v)$, and in expectation, at most $O\left(d \log \frac{|S|}{L_1(u,v)}\right)$ times $L_1(u, v)$.*

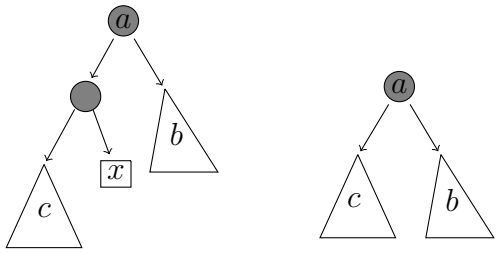
Theorem 4.2 provides a *low stretch distance preserving embedding*, reminiscent of the Johnson-Lindenstrauss Lemma [Johnson and Lindenstrauss, 1984] using random projections for $L_2()$ distances (which has much better dependence on $d$). The theorem is interesting because it implies that if a point is far from others (as is the case with anomalies) that it will continue to be at least as far in a random cut tree in expectation. The proof of Theorem 4.2 follows along the same lines of the proof of approximating finite metric spaces by a collection of trees [Charikar et al., 1998], and appears in the chapter appendix.

The theorem shows that if there is a lot of empty space around a point, i.e., $\gamma = \min_v L_1(u, v)$ is large, then we will isolate the point within $O(d \log |S|/\gamma)$ levels from the root. Moreover since for any $p \geq 1$, the $p$-normed distance satisfies $d^{1-1/p} L_p(u, v) \geq L_1(u, v) \geq L_p(u, v)$ and therefore the early isolation applies to all large $L_p()$ distances *simultaneously*. This provides us a pointer towards the success of the original isolation forest algorithm in low to moderate dimensional data, because $d$ is small and the probability of choosing a dimension is not as important if they are small in number. Thus the RRCF ensemble contains sufficient information that allows us to determine distance based anomalies, without focusing on the specifics of the distance function. Moreover the distance scales are adjusted appropriately based on the empty spaces between the points since the two bounding boxes may shrink after the cut.

Suppose that we are interested in the sample maintenance problem of producing a tree at random (with the correct probability) from $\mathcal{T}(S - \{x\})$ or from $\mathcal{T}(S \cup \{x\})$. In this paper we prove that we can efficiently insert and delete points into a random cut tree.

**Theorem 4.3** (Section 4.3)**.** *Given a tree $T$ drawn according to $\mathcal{T}(S)$; if we delete the node containing the isolated point $x$ and its parent (adjusting the grandparent accordingly, see Figure 4.2), then the resulting tree $T'$ has the same probability as if being drawn from $\mathcal{T}(S - \{x\})$. Likewise, we can produce a tree $T''$ as if drawn at random from $T(S \cup \{x\})$ is time which is $O(d)$ times the maximum depth of $T$, which is typically sublinear in $|T|$.*

Theorem 4.3 demonstrates an intuitively *natural* behavior when points are deleted — as shown in the schematic in Figure 4.1. In effect, if we insert $x$, perform a few more operations and then delete $x$, then not only do we preserve distributions but the trees remain very close to each other — as if the insertion never happened. This behavior is a classic desiderata of sketching algorithms.




(a) Before: $T$ (b) After: $T'$

Figure 4.1: Decremental maintenance of trees.

The natural behavior of deletions is not true if we do not choose the dimensions as in Step (1) of RRCF construction. For example, if we choose the dimensions uniformly at random as in [Liu et al., 2012], suppose we build a tree for $(1, 0), (\epsilon, \epsilon), (0, 1)$ where $1 \gg \epsilon > 0$ and then delete $(1, 0)$. The probability of getting a tree over the two remaining points that uses a vertical separator is $3/4 - \epsilon/2$ and not $1/2$ as desired. The probability of getting that tree in the RRCF process (after applying Theorem 4.3) is $1-\epsilon$, as desired. This natural behavior under deletions is also not true of most space partitioning methods –such as quadtrees [Finkel and Bentley, 1974], kd-trees [Bentley, 1975], and R-trees [Guttman, 1984]. The dynamic maintenance of a distribution over trees in a streaming setting is a novel contribution to the best of our knowledge and as a consequence, we can efficiently maintain a tree over a sample of a stream:

**Theorem 4.4.** *We can maintain a random tree over a sample $S$ even as the sample $S$ is updated dynamically for streaming data using sublinear update time and $O(d|S|)$ space.*

We can now use reservoir sampling [Vitter, 1985] to maintain a uniform random sample of size $|S|$ or a recency-biased weighted random sample of size $|S|$ [Efraimidis and Spirakis, 2006], in space proportional to $|S|$ on the fly. In effect, the random sampling process is now orthogonal from the robust random cut forest construction. For example to produce a sample of size $\rho|S|$ for $\rho < 1$, in an uniform random sampling we can perform straightforward rejection sampling; in the recency biased sample in [Efraimidis and Spirakis, 2006] we need to delete the $(1 - \rho)|S|$ lowest priority points. This notion of downsampling via deletions is supported perfectly by Theorem 4.3 – even for downsampling rates that are determined after the trees have been constructed, during postprocessing. Thus,

**Theorem 4.5.** *Given a tree $T(S)$ for sample $S$, if there exists a procedure that downsamples via deletion, then we have an algorithm that simultaneously provides us a downsampled tree for every downsampling rate.*

Theorems 4.4 and 4.5 taken together separate the notion of sampling from the analysis task and therefore eliminates the need to fine tune the sample size as an initial parameter. Moreover the dynamic maintenance of trees in Theorem 4.4 provides a mechanism to answer counterfactual questions as given in Theorem 4.6.

**Theorem 4.6.** *Given a tree $T(S)$ for sample $S$, and a point $p$ we can efficiently compute a random tree in $\mathcal{T}(S \cup \{p\})$, and therefore answer questions such as: what would have been the expected depth had $p$ been included in the sample?*

The ability to answer these counterfactual questions are critical to determining anomalies. Intuitively, we label a point $p$ as an anomaly when the joint distribution of including the point is significantly different from the distribution that excludes it. Theorem 4.6 allows us to efficiently (pretend) sketch the joint distribution including the point $p$. However instead of measuring the effect of the sampled data points on $p$ to determine its label (as is measured by notions such as expected depth), it stands to reason that we should measure the effect of $p$ on the sampled points. This leads us to the definition of anomalies used in this paper.

**Roadmap:** We discuss the decision procedure for defining anomalies in Section 4.2. In Section 4.3 we give algorithms to dynamically update RRCF trees through insertions and deletions. We discuss some of the less related unsupervised anomaly detection in Section 4.4. Finally, we validate the algorithms on publicly available real data demonstrating higher positive precision and positive recall in Section 4.5. We conclude with open directions of research in Section 4.6.

## 4.2 Defining Anomalies

OKKE SAYS: include the following paragraph? In a parametric setting, if we have a ← model of the distribution then one classic statistical approach can therefore be to use notions of divergences such as Kullback-Leibler or other Bregman divergences to measure the "surprise" generated by including a query point to a sample of data [Amari and Nagaoka, 2007]. However this classic approach is difficult to pursue unless we have large sample sizes to represent the density of multidimensional distributions. The classical approach corresponds to modeling the decision of including the observation versus the decision of not including the observation, in other words the externality imposed by the observation on the remainder of the data points. The task is therefore to define a non-parametric notion of externality. However externality alone will unlikely be the complete story, because for a typical (not computationally powerful) person to agree on an outlier, the person needs to find the outlier first. Consider the hypotheses:

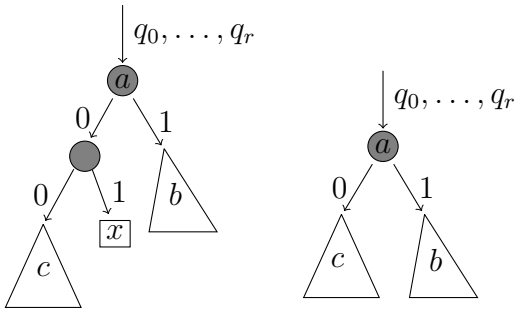
(a) An anomaly is often easy to describe – consider Waldo wearing a red fedora in a sea of dark felt hats. While it may be difficult for us to find Waldo in a crowd, if we could forget the faces and see the color (as is the case when Waldo is revealed by someone else) then the recognition of the anomaly is fairly simple.

(b) An anomaly makes it harder to describe the remainder of the data – if Waldo were not wearing the red fedora, we may not have admitted the possibility that hats can be colored. In essence, an anomaly displaces our **attention** from the normal observation to this new one.

The fundamental task is therefore to quantify the shift in attention. Suppose that we assign left branches the bit 0 and right branches the bit 1 in a tree in a random cut forest. Now consider the bits that specify a point (excluding the bits that are required to store the attribute values of the point itself). This defines the complexity of a random model $M_T$ which in our case corresponds to a tree $T$ that fits the initial data. Therefore the number of bits required to express a point corresponds to its depth in the tree.

Given a set of points $Z$ and a point $y \in Z$ let $f(y, Z, T)$ be the depth of $y$ in tree $T$. Consider now the tree produced by deleting $x$ as in Theorem 4.3 as $T(Z - \{x\})$. Note that given $T$ and $x$ the tree $T(Z - \{x\})$ is uniquely[2] determined. Let the depth of $y$ in $T(Z - \{x\})$ be $f(y, Z - \{x\}, T)$ (we drop the qualification of the tree in this notation since it is uniquely defined).

[2]The converse is not true, this is a many-to-one mapping.




(a) Tree $T(Z)$ (b) Tree $T(Z - \{x\})$

Figure 4.2: A correspondence of trees

Consider now a point $y$ in the subtree $c$ in Figure 4.2a. Its bit representation in $T$ would be $q_0, \ldots, q_r, 0, 0, \ldots$. The model complexity, denoted as $|M(T)|$ the number of bits required to write down the description of all points $y$ in tree $T$ therefore will be $|M(T)| = \sum_{y \in Z} f(y, Z, T)$. If we were to remove $x$ then the new model complexity is

$$|M(T')| = \sum_{y \in Z - \{x\}} f(y, Z - \{x\}, T')$$

where $T' = T(Z - \{x\})$ is a tree over $Z - \{x\}$. Now consider the expected **change** in model complexity under a random model.

$$\begin{aligned}
&\mathbb{E}_T \left[|M(T)|\right] - \mathbb{E}_{T(Z-\{x\})} \left[|M(T(Z - \{x\})|\right] \\
&= \sum_T \sum_{y \in Z} \mathbb{P}r \left[T\right] f(y, Z, T) \\
&- \sum_{T(Z-\{x\})} \sum_{y \in Z - \{x\}} \mathbb{P}r \left[T - \{x\}\right] f(y, Z - \{x\}, T)
\end{aligned}$$

However since we have a many to one mapping from $T(Z)$ to $T(Z - \{x\})$ as a consequence of Theorem 4.3, we can express the second sum over $T(Z)$ instead of $T' = T(Z - \{x\})$ and we get

$$\begin{aligned}
&\mathbb{E}_{T(Z)}\left[|M(T)|\right] - \mathbb{E}_{T(Z-\{x\})}\left[|M(T(Z-\{x\})|\right] \\
&= \sum_{T} \sum_{y \in Z-\{x\}} \mathbb{P}r\left[T\right] \Big( f(y, Z, T) - f(y, Z-\{x\}, T') \Big) \\
&+ \sum_{T} \mathbb{P}r\left[T\right] f(x, Z, T) \qquad (4.1)
\end{aligned}$$

**Definition 4.7.** *Define the bit-displacement or displacement of a point $x$ to be the increase in the model complexity of all other points, i.e., for a set $Z$, to capture the externality introduced by $x$, define, where $T' = T(Z - \{x\})$,*

$$\textsc{Disp}(x, Z) = \sum_{T, y \in Z-\{x\}} \mathbb{P}r\left[T\right] \Big( f(y, Z, T) - f(y, Z-\{x\}, T') \Big)$$

Note the total change in model complexity is $\textsc{Disp}(x, Z) + g(x, Z)$ where $g(x, Z) = \sum_T \mathbb{P}r\left[T\right] f(x, Z, T)$ is the expected depth of the point $x$ in a random model. Instead of postulating that anomalies correspond to large $g()$, we focus on larger values of $\textsc{Disp}()$. The name displacement is clearer based on this lemma:

**Lemma 4.8.** *The expected displacement caused by a point $x$ is the expected number of points in the sibling node of the leaf node containing $x$, when the partitioning is done according to the algorithm in Definition 4.1.*

*Proof.* In the absence of $x$, (in Figure 4.2b) the representation would be $q_0, \ldots, q_r, 0, \ldots$, in other words we would need 1 fewer bit to represent the point $p$. Therefore:

$$f(y, Z, T) - f(y, Z-\{x\}, T) = \begin{cases} 1 & y \in \text{sibling } c \text{ of } x \\ 0 & \text{otherwise} \end{cases}$$

The lemma follows. □

**Uniform Random Samples.** Since the dataset $Z$ is often large, it is unrealistic to have trees built over the entire dataset. However if we choose an uniform random sample $S$ of size $k$ and $\textsc{Disp}(x, Z)$ is large then,

$$\mathbb{E}_{S \subseteq Z, |S|=k}\left[\textsc{Disp}(x, S)\right] \approx (k/|Z|)\textsc{Disp}(x, Z)$$

with high probability. This is also a consequence of Theorem 4.5 and uniform random sampling from the set of nodes which are in the sibling node of the leaf node containing $x$.

**Shortcomings.** While Definition 4.7 points towards a possible definition of an anomaly, the definition as stated are not robust to duplicates or near-duplicates. Consider one dense cluster and a point $p$ far from away from the cluster. The displacement of $p$ will be large. But if there is a point $q$ very close to $p$, then $q$'s displacement in the presence of $p$ is small. This phenomenon is known as outlier masking. Duplicates and near duplicates are natural and therefore the semantics of any anomaly detection algorithm has to accommodate them.

**Duplicate Resilience.** Consider the notion that Waldo has a few friends who help him hide – these friends are **colluders**; and if we were to get rid of all the colluders then the description changes significantly. Specifically, instead of just removing the point $x$ we remove a set $C$ with $x \in C$. Analogous to Equation (4.1) ,

$$\begin{aligned} &\mathbb{E}_{T(Z)}\left[|M(T)|\right] - \mathbb{E}_{T(Z-C)}\left[|M(T(Z-C)|\right] \\ &\quad = \textsc{Disp}(C, Z) + \sum_{T} \sum_{y \in C} \mathbb{P}r\left[T\right] f(y, Z, T) \end{aligned} \tag{4.2}$$

where $\textsc{Disp}(C, Z)$ is the notion of displacement extended to subsets denoted as, where $T'' = T(Z - C)$,

$$\sum_{T, y \in Z - C} \mathbb{P}r\left[T\right] \Big( f(y, Z, T) - f(y, Z - C, T'') \Big) \tag{4.3}$$

Absent of any domain knowledge it appears that the displacement should be attributed equally to all the points in $C$. Therefore a natural choice of determining $C$ seems to be $\max \textsc{Disp}(C, Z)/|C|$ subject to $x \in C \subseteq Z$. However two problems arise. First there are too many subsets $C$, and second, in a streaming setting it is likely we would be using a sample $S \subset Z$. Therefore the supposedly natural choice does not extend to samples. To avoid both issues, we allow the choice of $C$ to be different for different samples $S$; in effect we are allowing Waldo to collude with different members in different tests! This motivates the following:

**Definition 4.9.** *The* **Collusive Displacement** *of $x$ denoted by* $\textsc{CoDisp}(x, Z, |S|)$ *of a point $x$ is defined as*

$$\mathop{\mathbb{E}}_{S \subseteq Z, T} \left[ \max_{x \in C \subseteq S} \frac{1}{|C|} \sum_{y \in S - C} \Big( f(y, S, T) - f(y, S - C, T'') \Big) \right]$$

**Lemma 4.10.** $\textsc{CoDisp}(x, Z, |S|)$ *can be estimated efficiently.*

*Proof.* Observe that following the logic of Lemma 4.8, the difference

$$f(y, S, T) - f(y, S - C, T)$$

is nonzero for $y \in Z - C$ if and only iff we delete all the elements in a sibling subtree containing $y$. For example in Figure 4.2a, if $|b|$ is large, then the collusive displacement will be large only if we delete all the nodes in $c$ along with $x$. Moreover to achieve any nonzero displacement we have to *simultaneously delete* all copies of $x$; and the result will scale down based on the number of duplicates. Observe that a consequence, given a $T$, we can compute

$$\max_{x \in C \subseteq S} \frac{1}{|C|} \sum_{y \in S - C} \Big( f(y, S, T) - f(y, S - C, T) \Big)$$

optimally by considering only the subtrees in the leaf to root path defined by $x$. □

While CoDisp$(x, Z, |S|)$ is dependent on $|S|$, the dependence is not severe. We envision using the largest sample size which is permitted under the resource constraints. We arrive at the central characterization we use in this paper:

**Definition 4.11.** *Outliers correspond to large* CoDisp().

## 4.3 Forest Maintenance on a Stream

In this section we discuss how Robust Random Cut Trees can be dynamically maintained. In the following, let $RRCF(S)$ be a the distribution over trees by running Definition 4.1 on $S$. Consider the following operations:

(i) **Insertion:** Given $T$ drawn from distribution $RRCF(S)$ and $p \notin S$ produce a $T'$ drawn from $RRCF(S \cup \{p\})$.

(ii) **Deletion:** Given $T$ drawn from distribution $RRCF(S)$ and $p \in S$ produce a $T'$ drawn from $RRCF(S - \{p\})$.

We need the following simple observation.

**Observation 4.12.** *Separating a point set $S$ and $p$ using an axis-parallel cut is possible if and only if it is possible to separate the minimal axis-aligned bounding box $B(S)$ and $p$ using an axis-parallel cut.*

The next lemma provides a structural property about RRCF trees. We are interest in incremental updates with as few changes as possible to a set of trees. Note that

given a specific tree we have two exhaustive cases, that (i) the new point which is to be deleted (respectively inserted) is not separated by the first cut and (ii) the new point is deleted (respective inserted) is separated by the first cut. Lemma 4.13 addresses these for collections of trees (not just a single tree) that satisfy (i) and (ii) respectively.

**Lemma 4.13.** *Given point $p$ and set of points $S$ with an axis parallel minimal bounding box $B(S)$ such that $p \notin B$:*

*(i) For any dimension $i$, the probability of choosing an axis parallel cut in a dimension $i$ that splits $S$ using the weighted isolation forest algorithm is exactly the same as the conditional probability of choosing an axis parallel cut that splits $S \cup \{p\}$ in dimension $i$, conditioned on not isolating $p$ from all points of $S$.*

*(ii) Given a random tree of $RRCF(S \cup \{p\})$, conditioned on the fact the first cut isolates $p$ from all points of $S$, the remainder of the tree is a random tree in $RRCF(S)$.*

*Proof.* Consider the first part. Let the length of the minimum bounding box of $S$ in dimension $i$ be $\ell_i$. Let the length of the minimum bounding box of $S \cup \{p\}$ in dimension $i$ be $\ell'_i$. Thus the probability (density) of choosing a cut $C$ in dimension $i$ that splits $S$ is

$$\frac{1}{\ell_i}\frac{\ell_i}{\sum_i \ell_i}$$

where the first term is the probability density conditioned on the dimension and the second term is the probability of choosing dimension $i$. The probability density of achieving the same cut in constructing a weighted isolation forest of $S \cup \{p\}$ conditioned on not isolating $p$ and $S$ is

$$\frac{1}{\ell_i}\mathbb{P}r\left[\text{Choosing dimension } i | \text{not isolating } p \text{ and } S\right]$$

Probability of choosing dimension $i$ **and** not isolating $p$ and $S$ is $\ell_i / \sum_i \ell'_i$. Therefore

$$\begin{aligned}
&\mathbb{P}r\left[\text{Choosing dimension } i | \text{not isolating } p \text{ and } S\right] \\
=&\frac{\mathbb{P}r\left[\text{Choosing dimension } i \textbf{ and } \text{not isolating } p \text{ and } S\right]}{\mathbb{P}r\left[\text{not isolating } p \text{ and } S\right]} \\
=&\frac{\ell_i / \sum_i \ell'_i}{\sum_i \ell_i / \sum_i \ell'_i} = \frac{\ell_i}{\sum_i \ell_i}
\end{aligned}$$

The last part follows from the observation that the probability of not isolating $p$ and $S$ is $\sum_i \ell_i / \sum_i \ell'_i$. This proves part (i). Note that part (ii) follows from construction. □

### 4.3.1 Deletion of Points

We begin with Algorithm 4 which is deceptively simple.

**Algorithm 4** Algorithm ForgetPoint.

1: Find the node $v$ in the tree where $p$ is isolated in $T$.
2: Let $u$ be the sibling of $v$. Delete the parent of $v$ (and of $u$) and replace that parent with $u$ (i.e., we short circuit the path from $u$ to the root).
3: Update all bounding boxes starting from $u$'s (new) parent upwards – this state is not necessary for deletions, but is useful for insertions.
4: Return the modified tree $T'$.

We now indicate how the above algorithm follows from Lemma 4.13.

**Lemma 4.14.** *If $T$ were drawn from the distribution $RRCF(S)$ then Algorithm 4 produces a tree $T'$ which is drawn at random from the probability distribution $RRCF(S - \{p\})$.*

*Proof.* Given $T$ was drawn from $RRCF(S)$ consider the random forest algorithm that would have produced this tree. Consider also the random forest algorithm as it produces $T'$. We will stochastically couple the decisions of the split operation – mirror the same split in $T'$ as in $T$ [Lindvall, 1992]. Even though the splits across the two trees are correlated, if we consider only $T$ or $T'$, it would appear that the respective tree was produced with the right distribution. Of course the mirroring will not always be obvious, but we address that below. Initially we have a set $S' = S$. Consider the cases (a)–(b) below:

(a) Suppose that we choose the dimension $i$ in splitting $T$ and the point $p$ does not lie on the bounding box of $S'$ in dimension $i$. In this case the presence or absence of the point $p$ does not affect the distribution of cuts are the same irrespective of the point set $S'$ or $S' - \{p\}$. The construction of $T'$ therefore can choose the same dimension $i$ and the same cut as in $T$, and that could correspond to be a valid step with the correct probability. Note that after the cut, $p$ can belong to only one side ,say $S''$. We will set $S' = S''$ and recurse. Note that we can use the same subtree (as in $T$) for the subset $S' - S''$ since there were no change to the point set. The construction of $T'$ can completely mirror $T$ for these subsets, and the construction will preserve the correct probabilities.

(b) Otherwise, we choose dimension $i$ in splitting $T$ and point $p$ lies on the bounding box of $S'$ in dimension $i$. We now have two cases.

(i) Point $p$ is separated from rest of $S'$. In this case $T$ produces a sibling tree $T(u)$ starting at node $u$ which is the sibling node of node $v$ containing the isolated point $p$. But then in $T'$ we do not have $p$, and $T(u)$ is a random tree from $RRCF(S' - \{p\})$ and the construction is correct using part (ii) of Lemma 4.13.

(ii) Point $p$ is not separated from $S'$. By Lemma 4.13, conditioned on the fact that $p$ is not separated from $S'$ we are choosing a random cut which separates $S' - \{p\}$ along a chosen dimension $i$. This is therefore an appropriate choice for $T'$ using part (i) of Lemma 4.13 and we choose the same cut in $T'$. Again we have two subsets, and in $T$, $p$ belongs to only one side. We recurse on that side – for the other side the construction of $T, T'$ can be identical since they have the same set of points.

The above cases are mutually exclusive and exhaustive. This proves the Lemma. □

The next lemma is immediate.

**Lemma 4.15.** *The deletion operation can be performed in time $O(d)$ times the depth of point $p$.*

Observe that if we delete a random point from the tree, then the running time of the deletion operation is $O(d)$ times the expected depth of any point. Likewise if we delete points whose depth is shallower than most points in the tree then we can improve the running time of Lemma 4.15.

### 4.3.2 Insertion of Points

Given a tree $T$ from $RRCF(S)$ we produce a tree $T'$ from the distribution $RRCF(S \cup \{p\})$. The algorithm is provided in Algorithm 5. Once again we will couple the decisions that is mirror the same split in $T'$ as in $T$, as long as $p$ is not outside a bounding box in $T$. Up to this point we are performing the same steps as in the construction of the forest on $S \cup \{p\}$, with the same probability.

**Lemma 4.16.** *If $T$ were drawn from the distribution $RRCF(S)$ then Algorithm 4 produces a tree $T'$ which is drawn at random from the probability distribution $RRCF(S \cup \{p\})$.*

*Proof.* We proceed in a manner similar to the proof of Lemma 4.14 — *however instead of using tree $T$ to define the splits for $T'$, we will first make a decision about $T'$ and*

**Algorithm 5** Algorithm InsertPoint.

1: We have a set of points $S'$ and a tree $T(S')$. We want to insert $p$ and produce tree $T'(S' \cup \{p\})$.
2: If $S' = \emptyset$ then we return a node containing the single node $p$.
3: Otherwise $S'$ has a bounding box $B(S') = [x_1^\ell, x_1^h] \times [x_2^\ell, x_2^h] \times \cdots [x_d^\ell, x_d^h]$. Let $x_i^\ell \leq x_i^h$ for all $i$.
4: For all $i$ let $\hat{x}_i^\ell = \min\{p_i, x_i^\ell\}$ and $\hat{x}_i^h = \max\{x_i^h, p_i\}$.
5: Choose a random number $r \in [0, \sum_i (\hat{x}_i^h - \hat{x}_i^\ell)]$.
6: This $r$ corresponds to a specific choice of a cut in the construction of $RRCF(S' \cup \{p\})$. For instance we can compute $\arg\min\{j | \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) \geq r\}$ and the cut corresponds to choosing $\hat{x}_j^\ell + \sum_{i=1}^{j} (\hat{x}_i^h - \hat{x}_i^\ell) - r$ in dimension $j$.
7: If this cut separates $S'$ and $p$ (i.e., is not in the interval $[x_j^\ell, x_j^h]$) then and we can use this as the first cut for $T'(S' \cup \{p\})$. We create a node – one side of the cut is $p$ and the other side of the node is the tree $T(S')$.
8: If this cut does not separate $S'$ and $p$ then we throw away the cut! We choose the exact same dimension as $T(S')$ in $T'(S' \cup \{p\})$ and the exact same value of the cut chosen by $T(S')$ and perform the split. The point $p$ goes to one of the sides, say with subset $S''$. We repeat this procedure with a smaller bounding box $B(S'')$ of $S''$. For the other side we use the same subtree as in $T(S')$.
9: In either case we update the bounding box of $T'$.

*then mirror* $T$. Suppose that we have currently the set of nodes $S'$. Note that the case of $S' = \emptyset$ is trivial. Therefore assume $S' \neq \emptyset$ and we are given a tree $T(S')$ from $RRCF(S')$.

(a) If we decide to separate $p$ and $S'$ then Step 6 in Algorithm 5 generates such a cut. Now after the cut, we observe that $T(S')$ is already a random tree in $RRCF(S')$ and we can simply use $T(S')$ to define $T'$.

(b) If we decide to **not** separate $p$ and $S'$ then using part (i) of Lemma 4.13, we can choose any cut that splits the bounding box $B(S')$. *Note that the first cut in* $T(S')$ *is exactly such a cut chosen with the correct distribution.* Therefore we can use the same cut in $T'$ in this case. Note that we now have two sides and $p$ only affects one side – we can use the same subtree as in $T(S')$ for the side that does not contain $p$. We the side $S''$ that contains $p$. Note that we recursively maintain the property that we have a random tree $T(S'')$ from $RRCF(S'')$.

The above steps are mutually exclusive and exhaustive. This proves the Lemma. □

## 4.4 Isolation Forest and Other Related Work

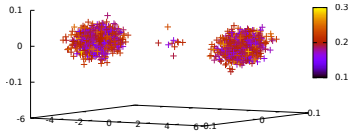
### 4.4.1 The Isolation Forest Algorithm

Recall that the isolation forest algorithm uses an ensemble of trees similar to those constructed in Definition 4.1, with the modification that the dimension to cut is chosen uniformly at random. Given a new point $p$, that algorithm follows the cuts and compute the average depth of the point across a collection of trees. The point is labeled an anomaly if the score exceeds a threshold; which corresponds to average depth being small compared to $\log |S|$ where $S$ is suitably sized sample of the data.

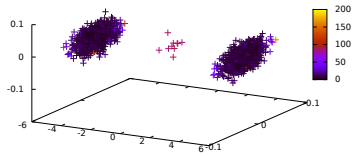
The advantage of the isolation forest is that different dimensions are treated independently and the algorithm is invariant to scaling different dimensions differently. However consider the following example.

**Example 4.17** (IRRELEVANT DIMENSIONS.)**.** *Suppose we have two clusters of* 1000 *points each corresponding to* $x_1 = \pm 5$ *in the first dimension, and* $x_i = 0$ *in all remaining dimensions* $i$*. In all coordinates (including* $x_1$*) we add a random Gaussian noise with mean* 0 *and standard deviation* 0.01 *simulating white noise. Now consider* 10 *points with* $x_1 = 0$ *and the same behavior in all the other coordinates. When* $d = 2$ *the small cluster of points in the center is easily separated by the isolation forest algorithm which treats the dimensions independently. When* $d = 30$ *the vast majority of cuts are in irrelevant dimensions, and the algorithm fails (when run on entire data) as shown in Figure 4.1a for a single trial over* 100 *trees. For* 10 *trials (for the same data set), the algorithm determined that* 430*,* 270*,* 147*,* 220*,* 48*,* 244*,* 193,158*,* 250 *and* 103 *points had the same of higher anomaly score than the point with the highest anomaly score among the* 10 *points (the identity of this point varied across the trials).*

In essence, the algorithm either produces too many false alarms or does not have good recall. Note that AUC is not a relevant measure here since the class sizes between anomalous and non-anomalous are skewed, $1 : 200$. The results were consistent across multiple data sets generated according to the example. Figure 4.3b shows a corresponding single trial using CoDisp(). The CoDisp() measure places the 10 points in the largest 20 values most of the time. Example 1 shows that scale independence therefore can be negative feature if distance is a meaningful concept in the dataset. However in many tasks that depend on detecting anomalies, the relevance of different dimensions is often unknown. The question of determining the appropriate scale of measurement often has far reaching consequences in data analysis.



(a) Performance of Isolation Forest [Liu et al., 2012]. Note that the score never exceeds 0.3 whereas a score of 0.5 corresponds to an outlier. Note also that the two clusters are not distinguishable from the 10 points near origin outliers in depth values (color).



(b) Performance of CoDisp$(x, Z, |Z|)$. Observe that the clusters and outliers are separated; some of the extremal points in the clusters have the same (collusive) displacement as the 10 points near the origin, which is expected.

Figure 4.3: The result of running isolation forest and CoDisp() on the input in Example 4.17 for $d = 30$.

A modified version of the above example also is helpful in arguing why depth of a point is a not always helpful in characterizing anomalies, even in low dimensions. Consider,

**Example 4.18** (Held Out Data.)**.** *Consider the same dataset as in Example 4.17 in* $d = 2$ *dimensions. Suppose that we have only sampled* 100 *points and all the samples correspond to* $x_1 = \pm 5$*. Suppose we now want to evaluate: is the point* $(0, 0)$ *an anomaly? Based on the samples the natural answer is yes. The scoring mechanism of isolation forest algorithm fails because once the two clusters are separated, this new point* $(0, 0)$ *behaves as a point in one of the two other clusters! The situation however changes completely if we include* $(0, 0)$ *to build the trees.*

The example explains why the isolation forest algorithm is sensitive to sample size. However most anomalies are not usually seen in samples – anomaly detection algorithms should be measured on held out data. Note that Theorem 4.6 can efficiently solve the issue raised in Example 4.18 by answering the counterfactual question of what is the expected height has we observed $(0, 0)$ in the sample (without rebuilding the trees). However expected depth seems to generate more false alarms, and we investigate this further in the supplementary material of [Guha et al., 2016].

### 4.4.2 Other Related Work

The problem of (unsupervised) outlier detection has a rich literature. We survey some of the work here; for an extensive survey see [Aggarwal, 2013, Chandola et al., 2009] and references therein. We discuss some of techniques which are unrelated to the concepts already discussed.

Perhaps the most obvious definition of an anomaly is density based outlier detection, which posits that a low-probability events are likely anomalous. This has led to different approaches based on estimating the density of data sets. For points in $\mathcal{R}^n$, Knorr and Ng [1997, 1998, 1999], Knorr et al. [2000] estimate the density by looking at the number of points that are within a ball of radius $d$ of a given data point. The lower this number, the more anomalous the data point is. This approach may break down when different parts of the domain have different scales. To remedy this, there a methods [Breunig et al., 1999, 2000] that look at the density around a data point compared to its neighborhood. A variation of the previous approach is to consider a fixed $k$ number of nearest neighbors and base the anomaly score on this [Eskin et al., 2002, Zhang and Wang, 2006]. Here the anomaly score is monotonically increasing in the distances to the $k$ nearest-neighbors. Taking the idea of density one step further, some authors have looked at finding structure in the data through clustering. The intuition here is that for points that cannot easily be assigned to a cluster, there is no good explanation for their existence. There are several clustering algorithms that work well to cluster part of the data, such as DBSCAN [Ester et al., 1996] and STREAM [Guha et al., 2003]. Additionally, FindOut [Yu et al., 2002] removes points it cannot cluster, and then recurses. Finally the notion of sketching used in this paper is orthogonal to the notion used in [Huang and Kasiviswanathan, 2015] which uses streaming low rank approximation of the data.

## 4.5 Experiments

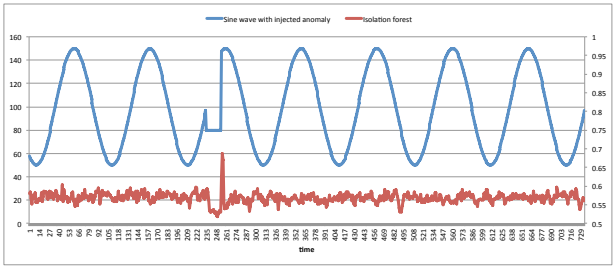
In the experiments, we focus on datasets where anomalies are visual, verifiable and interpretable. We begin with a synthetic dataset that captures the classic diurnal rhythm of human activity. We then move to a real dataset reflecting taxi ridership in New York City. In both cases, we compare the performance of RRCF with IF.

A technique that turns out to be useful for detecting anomalies in streams is shingling. If a shingle of size 4 is passed over a stream, the first 4 values of the stream received at time $t_1, t_2, t_3, t_4$ are treated as a 4-dimensional point. Then, at time $t_5$, the values at time $t_2, t_3, t_4, t_5$ are treated as as the next four-dimensional point. The window slides over one unit at each time step. A shingle encapsulates a typical shape of a curve – a departure from a typical shape could be an anomaly.

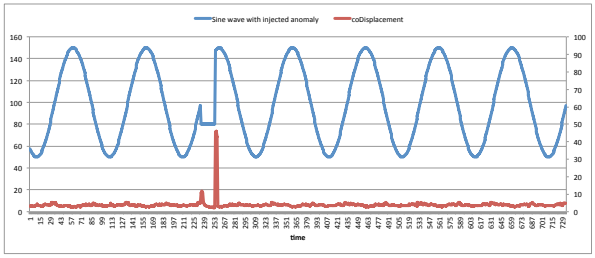
### 4.5.1 Synthetic Data

Many real datasets implicitly reflect human circadian rhythms. For example, an eCommerce site may monitor the number of orders it receives per hour. Search engines may monitor search queries or ad clicks per minute. Content delivery networks may monitor requests per minute. In these cases, there is a natural tendency to expect higher values during the day and lower values at night. An anomaly may reflect an unexpected dip or spike in activity.

In order to test our algorithm, we synthetically generated a sine wave where a dip is artificially injected around timestamp 500 that lasts for 20 time units. The goal is to determine if our anomaly detection algorithm can spot the beginning and end of the injected anomaly. The experiments were run with a shingle of length four, and one hundred trees in the forest, where each tree is constructed with a uniform random reservoir sample of 256 points. We treat the dataset as a stream, scoring a new point at time $t + 1$ with the data structure built up until time $t$.




(a) The bottom red curve reflects the anomaly score produced by IF. Note that the start of the anomaly is missed.




(b) The bottom red curve represents the anomaly score produced by RRCF. Both the beginning and end of the anomaly are caught.

Figure 4.4: The top blue curve represents a sine wave with an artificially injected anomaly. The bottom red curve shows the anomaly score over time.

Table 4.1: Comparison of Baseline Isolation Forest to proposed Robust Random Cut Forest

| Method | Sample Size | Positive Precision | Positive Recall | Negative Precision | Negative Recall | Accuracy | AUC |
|---|---|---|---|---|---|---|---|
| IF | 256 | 0.42 (0.05) | 0.37 (0.02) | 0.96 (0.00) | 0.97 (0.01) | 0.93 (0.01) | 0.83 (0.01) |
| RRCF | 256 | 0.87 (0.02) | 0.44 (0.04) | 0.97 (0.00) | 1.00 (0.00) | 0.96 (0.00) | 0.86 (0.00) |
| IF | 512 | 0.48 (0.05) | 0.37 (0.01) | 0.97 (0.01) | 0.96 (0.00) | 0.94 (0.00) | 0.86 (0.00) |
| RRCF | 512 | 0.84 (0.04) | 0.50 (0.03) | 0.99 (0.00) | 0.97 (0.00) | 0.96 (0.00) | 0.89 (0.00) |
| IF | 1024 | 0.51 (0.03) | 0.37 (0.01) | 0.96 (0.00) | 0.98 (0.00) | 0.94 (0.00) | 0.87 (0.00) |
| RRCF | 1024 | 0.77 (0.03) | 0.57 (0.02) | 0.97 (0.00) | 0.99 (0.00) | 0.96 (0.00) | 0.90 (0.00) |

Table 4.2: Segment-Level Metrics and Precision@K

| Method | Segment Precision | Segment Recall | Time to Detect Onset | Time to Detect End | Prec@5 | Prec@10 | Prec@15 | Prec@20 |
|---|---|---|---|---|---|---|---|---|
| IF | 0.40 (0.09) | 0.80 (0.09) | 22.68 (3.05) | 23.30 (1.54) | 0.52 (0.10) | 0.50 (0.00) | 0.34 (0.02) | 0.28 (0.03) |
| RRCF | 0.65 (0.14) | 0.80 (0.00) | 13.53 (2.05) | 10.85 (3.89) | 0.58 (0.06) | 0.49 (0.03) | 0.39 (0.02) | 0.30 (0.00) |

In Figure 4.4a, we show the result of running IF on the sine wave. For anomalies, detecting the onset is critical – and even more important than detecting the end of an anomaly. Note that IF misses the start of the anomaly at time 500. The end of the anomaly is detected, however, by then the system has come back to its normal state – it is not useful to fire an alarm once the anomaly has ended. Next, consider Figure 4.4b which shows the result of running RRCF on the same sine wave. Observe that the two highest scoring moments in the stream are the end and the beginning of the anomaly. The anomaly is successfully detected by RRCF. While the result of only a single run is shown, the experiment was repeated many times and the picture shown in Figure 4.4 is consistent across all runs.

### 4.5.2 Real Life Data: NYC Taxicabs

Next we conduct a streaming experiment using taxi ridership data from the NYC Taxi Commission[3]. We consider a stream of the total number of passengers aggregated over a 30 minute time window. Data is collected over a 7-month time period from 7/14 – 1/15. Note while this is a 1-dimensional datasets, we treat it as a 48-dimensional data set where each point in the stream is represented by a sliding window or shingle of the last day of data, ignoring the first day of data. The intuition is that the last day of activity captures a typical shape of passenger ridership.

The following dates were manually labeled as anomalies based on knowledge of holidays and events in NYC [Lavin and Ahmad, 2015]: Independence Day (7/4/14-7/6/14), Labor Day (9/1/14), Labor Day Parade (9/6/14), NYC Marathon (11/02/14),

[3]http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml

Thanksgiving (11/27/14), Christmas (12/25/14), New Years Day (1/1/15), North American Blizzard (1/26/15-1/27/15). For simplicity, we label a 30-minute window an anomaly if it overlaps one of these days.

**Stream** We treat the data as a stream – after observing points $1, \ldots, i$, our goal is to score the $(i+1)$st point. The score that we produce for $(i+1)$ is based only on the previous data points $1, \ldots, i$, but not their labels. We use IF as the baseline. While a streaming version was subsequently published [Tan et al., 2011], since it was not found to improve over IF [Emmott et al., 2013], we consider a more straightforward adaptation. Since each tree in the forest is created based on a random sample of data, we simply build each tree based on a random sample of the stream, e.g., uniform or time-decayed as previously referenced. Our aim here is to compare to the baseline with respect to accuracy, not running time. Each tree can be updated in an embarrassingly parallel manner for a faster implementation.

**Metrics** To quantitatively evaluate our approach, we report on a number of precision/recall-related metrics. We learn a threshold for a good score on a training set and report the effectiveness on a held out test set. The training set contains all points before time $t$ and the test set all points after time $t$. The threshold is chosen to optimize the F1-measure (harmonic mean of precision and recall). We focus our attention on positive precision and positive recall to avoid "boy who cried wolf" effects [Tsien and Fackler, 1997, Lawless, 1994].

For the finer granularity data in the taxi cab data set, we view the ground truth as segments of time when the data is in an anomalous state. Our goal is to quickly and reliably identify these segments. We say that a segment is identified in the test set if the algorithm produces a score over the learned threshold anytime during the segment (including the sliding window, if applicable).

**Results** In the experiments, there were 200 trees in the forest, each computed based on a random sample of 1K points. Note that varying the sample size does not alter the nature of our conclusions. Since ridership today is likely similar to ridership tomorrow, we set our time-decayed sampling parameter to the last two months of ridership. All results are averaged over multiple runs (10). Standard deviation is also reported. Figure 4.5 shows the result of the anomaly scores returned by CoDisp().

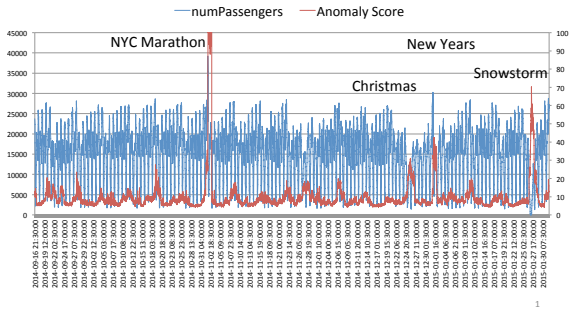



Figure 4.5: NYC taxi data and CoDisp(). Note that Thanksgiving is not captured.

In a more detailed evaluation, the first set of results (Table 4.1) show that the proposed RRCF method is more accurate than the baseline. Particularly noteworthy is RRCF's higher positive precision, which implies a lower false alarm rate. In Table 4.2, we show the segment-based results. Whereas Table 4.1 may give more credit for catching a long anomaly over a short one, the segment metric weighs each alarm equally. The proposed RRF method not only catches more alarms, but also catches them more quickly. The units are measured in 30 minute increments – so 11 hours on average to catch an alarm on the baseline and 7 hours for the RRCF method. These actual numbers are not as important here, since anomaly start/end times are labeled somewhat loosely. The difference in time to catch does matter. Precision@K is also reported in Table 4.2.

**Discussion:** Shingle size, if used, matters in the sense that shingles that are too small may catch naturally varying noise in the signal and trigger false alarms. On the other hand, shingles that are too large may increase the time it takes to find an alarm, or miss the alarm altogether. Time decay requires knowledge of the domain. Sample size choice had less effect – with varying sample sizes of 256, 512 and 1K the conclusions are unchanged on this dataset.

## 4.6 Conclusions and Future Work

We introduced the robust random cut forest sketch and proved that it approximately preserves pairwise distances. If the data is recorded in the correct scale, distance is crucially important to preserve for computations, and not just anomaly detection. We adopted a model-based definition of an anomaly that captures the differential effect of adding/removing a point on the size of the sketch. Experiments suggest that the algorithm holds great promise for fighting alarm fatigue as well as catching more missed alarms.

We believe that the random cut forest sketch is more beneficial than what we

have established. For example, it may also be helpful for clustering since pairwise distances are approximately preserved. In addition, it may help detect changepoints in a stream. A *changepoint* is a moment in time $t$ where before time $t$ the data is drawn from a distribution $D_1$ and after time $t$ the data is drawn from a distribution $D_2$, and $D_1$ is sufficiently different from $D_2$ [Kifer et al., 2004, Dasu et al., 2006]. By maintaining a sequence of sketches over time, one may be able to compare two sketches to determine if the distribution has changed.

# Chapter Appendix

## 4.A Proof of Theorem 4.2

Observe that in the construction of $RRCF(S)$ where the sum of the lengths of the sides of $B(S)$ is denoted by $P(S)$, the probability that we separated two points $\mathbf{x}(1), \mathbf{x}(2)$ by the first cut is proportional to the $L_1$ distance measure:

$$\frac{1}{P(S)} \sum_i |\mathbf{x}(1)_i - \mathbf{x}(2)_i|$$

Also observe that $P(S)$ decreases by at least a factor of $(1 - \frac{1}{2d})$. To observe this, note that if we start with a rectangle where side $i$ has length $\ell_i$ (and $\sum_i \ell_i = P(S)$ then the new expected perimeter of either side decreases by (the first term corresponds to choosing the dimension and the second term is the expected decrease in that dimension),

$$\sum_i \frac{\ell_i}{P(S)} \frac{\ell_i}{2} \geq \frac{\sum_i \ell_i^2}{2 \sum_i \ell_i} \geq \frac{\sum_i \ell_i}{2d}$$

**Lemma 4.19.** *Given a tree $T$ in $RRCF(S)$ suppose we measure the distance between two points as follows: we find the first level where the points are separated and let the set be $S$. We assign the distance $P(S)$, i.e., the sum of the edge lengths of the minimum bounding box $B(S)$. Then the expected length of pair of points $\mathbf{x}(1), \mathbf{x}(2)$ is $L_1(\mathbf{x}(1), \mathbf{x}(2))$ times the expected number of steps to separate $\mathbf{x}(1), \mathbf{x}(2)$. Observe that we never assign a distance less than $L_1(\mathbf{x}(1), \mathbf{x}(2))$.*

*Proof.* Follows from the fact that the distance assigned at a level corresponding to $S'$ is $P(S')$ and the probability of that distance assignment is $L_1(\mathbf{x}(1), \mathbf{x}(2))/P(S')$. The expected distance therefore is $L_1(\mathbf{x}(1), \mathbf{x}(2))$ times the expected number of steps that separate the two points! □

**Remark:** Note that the expected number of steps can be bounded by $O(d \log P(S)/L_1)$ since $P(S)$ decreases by a $(1 - \frac{1}{2d})$ factor in expectation. Other bounds can also be used – for example logarithm of the ratio of the total volume to the volume of the

smallest box that contains $\mathbf{x}(1), \mathbf{x}(2)$ since in each step we divide the volume by $\frac{1}{2}$ in expectation.

# Part II

# Incentives

# Chapter 5

# Online Prediction with Selfish Experts

Up until now, we have looked at the role that data can play in game theory.[1] This chapter still concerns learning from data, but it looks at how incentives may play a role in a learning setting that has thus far been incognizant of incentive issues. We consider the problem of binary prediction with expert advice in settings where experts have agency and seek to maximize their credibility. This chapter makes three main contributions. First, it defines a model to reason formally about settings with selfish experts, and demonstrates that "incentive compatible" (IC) algorithms are closely related to the design of proper scoring rules. Designing a good IC algorithm is easy if the designer's loss function is quadratic, but for other loss functions, novel techniques are required. Second, we design IC algorithms with good performance guarantees for the absolute loss function. Third, we give a formal separation between the power of online prediction with selfish experts and online prediction with honest experts by proving lower bounds for both IC and non-IC algorithms. In particular, with selfish experts and the absolute loss function, there is no (randomized) algorithm for online prediction—IC or otherwise—with asymptotically vanishing regret.

## 5.1 Introduction

In the months leading up to elections and referendums, a plethora of pollsters try to figure out how the electorate is going to vote. Different pollsters use different methodologies, reach different people, and may have sources of random errors, so generally the polls don't fully agree with each other. Aggregators such as Nate Silver's FiveThirtyEight, The Upshot by the New York Times, and HuffPost Pollster by

[1] This chapter is based on joint work with Tim Roughgarden [Roughgarden and Schrijvers, 2017].

the Huffington Post consolidate these different reports into a single prediction, and hopefully reduce random errors.[2] FiveThirtyEight in particular has a solid track record for their predictions, and as they are transparent about their methodology we use them as a motivating example in this paper. To a first-order approximation, they operate as follows: first they take the predictions of all the different pollsters, then they assign a weight to each of the pollsters based on past performance (and other factors), and finally they use the weighted average of the pollsters to run simulations and make their own prediction.[3]

But could the presence of an institution that rates pollsters inadvertently create perverse incentives for the pollsters? The FiveThirtyEight pollster ratings are publicly available.[4] The ratings can be interpreted as a reputation, and a low rating can negatively impact future revenue opportunities for a pollster. Moreover, it has been demonstrated in practice that experts do not always report their true beliefs about future events. For example, in weather forecasting there is a known "wet bias," where consumer-facing weather forecasters deliberately *over*estimate low chances of rain (e.g. a 5% chance of rain is reported as a 25% chance of rain) because people don't like to be surprised by rain [Bickel and Kim, 2008].

These examples motivate the development of models of aggregating predictions that endow agency to the data sources.[5] While there are multiple models in which we can investigate this issue, a natural candidate is the problem of prediction with expert advice. By focusing on a standard model, we abstract away from the fine details of FiveThirtyEight (which are anyways changing all the time), which allows us to formulate a general model of prediction with experts that clearly illustrates why incentives matter. In the classical model [Littlestone and Warmuth, 1994, Freund and Schapire, 1997], at each time step, several experts make predictions about an unknown event. An online prediction algorithm aggregates experts' opinions and makes its own prediction at each time step. After this prediction, the event at this time step is realized and the algorithm incurs a loss as a function of its prediction and the realization. To compare its performance against individual experts, for each

[2]FiveThirtyEight: `https://fivethirtyeight.com/`, The Upshot: `https://www.nytimes.com/section/upshot`, HuffPost Pollster: `http://elections.huffingtonpost.com/pollster`.

[3]This is of course a simplification. FiveThirtyEight also uses features like the change in a poll over time, the state of the economy, and correlations between states. See `https://fivethirtyeight.com/features/how-fivethirtyeight-calculates-pollster-ratings/` for details. Our goal in this paper is not to accurately model all of the fine details of FiveThirtyEight (which are anyways changing all the time). Rather, it is to formulate a general model of prediction with experts that clearly illustrates why incentives matter.

[4]`https://projects.fivethirtyeight.com/pollster-ratings/`

[5]More generally, one can investigate how the presence of machine learning algorithms affects data generating processes, either during learning, e.g. [Dekel et al., 2010, Cai et al., 2015], or during deployment, e.g. [Hardt et al., 2016, Brückner and Scheffer, 2011]. We discuss some of this work in the related work section.

expert the algorithm calculates what its loss would have been had it always followed the expert's prediction. While the problems introduced in this paper are relevant for general online prediction, to focus on the most interesting issues we concentrate on the case of binary events, and real-valued predictions in $[0, 1]$. For different applications, different notions of loss are appropriate, so we parameterize the model by a loss function $\ell$. Thus our formal model is: at each time step $t = 1, 2, \ldots, T$:

1. Each expert $i$ makes a prediction $p_i^{(t)} \in [0, 1]$, with higher values indicating stronger advocacy for the event "1."

2. The online algorithm commits to a probability distribution over $\{0, 1\}$, with $q^{(t)}$ denoting the probability assigned to "1."

3. The outcome $r^{(t)} \in \{0, 1\}$ is realized.

4. The algorithm incurs a loss of $\ell(q^{(t)}, r^{(t)})$ and calculates for each expert $i$ a loss of $\ell(p_i^{(t)}, r^{(t)})$.

The standard goal in this problem is to design an online prediction algorithm that is guaranteed to have expected loss not much larger than that incurred by the best expert in hindsight. The classical solutions maintain a weight for each expert and make a prediction according to which outcome has more expert weight behind it. An expert's weight can be interpreted as a measure of its credibility in light of its past performance. The (deterministic) Weighted Majority (WM) algorithm always chooses the outcome with more expert weight. The Randomized Weighted Majority (RWM) algorithm randomizes between the two outcomes with probability proportional to their total expert weights. The most common method of updating experts' weights is via multiplication by $1 - \eta\ell(p_i^{(t)}, r^{(t)})$ after each time step $t$, where $\eta$ is the learning rate. We call this the "standard" or "classical" version of the WM and RWM algorithm.

The classical model instills no agency in the experts. To account for this, in this paper we replace Step 1 of the classical model by:

1a. Each expert $i$ formulates a belief $b_i^{(t)} \in [0, 1]$.

1b. Each expert $i$ reports a prediction $p_i^{(t)} \in [0, 1]$ to the algorithm.

Each expert now has two types of loss at each time step — the *reported loss* $\ell(p_i^{(t)}, r^{(t)})$ with respect to the reported prediction and the *true loss* $\ell(b_i^{(t)}, r^{(t)})$ with respect to her true beliefs.[6]

[6]When we speak of the best expert in hindsight, we are always referring to the true losses. Guarantees with respect to reported losses follow from standard results [Littlestone and Warmuth, 1994, Freund and Schapire, 1997, Cesa-Bianchi et al., 2007], but are not immediately meaningful.

When experts care about the weight that they are assigned, and with it their reputation and influence in the algorithm, different loss functions can lead to different expert behaviors. For example, in Section 5.2 we observe that for the quadratic loss function, in the standard WM and RWM algorithms, experts have no reason to misreport their beliefs. The next example shows that this is not the case for other loss functions, such as the absolute loss function.[7]

**Example 5.1.** *Consider the standard WM algorithm, where each expert initially has unit weight, and an expert's weight is multiplied by* $1 - \eta|p_i^{(t)} - r^{(t)}|$ *at a time step* $t$*, where* $\eta \in (0, \frac{1}{2})$ *is the learning rate. Suppose there are two experts and* $T = 1$*, and that* $b_1^{(1)} = .49$ *while* $b_2^{(1)} = 1$*. Each expert reports to maximize her expected weight. Expanding, for each* $i = 1, 2$ *we have*

$$\begin{aligned}
\mathbb{E}[w_i^{(1)}] &= \Pr(r^{(1)} = 1) \cdot (1 - \eta(1 - p_i^{(1)})) + \Pr(r^{(1)} = 0) \cdot (1 - \eta p_i^{(1)}) \\
&= b_i^{(1)} \cdot (1 - \eta(1 - p_i^{(1)})) + (1 - b_i^{(1)}) \cdot (1 - \eta p_i^{(1)}) \\
&= b_i^{(1)} - b_i^{(1)}\eta + b_i^{(1)}\eta p_i^{(1)} + 1 - \eta p_i^{(1)} - b_i^{(1)} + b_i^{(1)}\eta p_i^{(1)} \\
&= 2b_i^{(1)}\eta p_i^{(1)} - p_i^{(1)}\eta - b_i^{(1)}\eta + 1,
\end{aligned}$$

*where all expectations and probabilities are with respect to the true beliefs of agent* $i$*. To maximize this expected weight over the possible reports* $p_i^{(1)} \in [0, 1]$*, we can omit the second two terms (which are independent of* $p_i^{(1)}$*) and divide out by* $\eta$ *to obtain*

$$\begin{aligned}
\arg\max_{p_i^{(1)} \in [0,1]} 2b_i^{(1)}\eta p_i^{(1)} - p_i^{(1)}\eta - b_i^{(1)}\eta + 1 &= \arg\max_{p_i^{(1)} \in [0,1]} p_i^{(1)}(2b_i^{(1)} - 1) \\
&= \begin{cases} 1 & \text{if } b_i^{(1)} \geq \frac{1}{2} \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}$$

*Thus an expert always reports a point mass on whichever outcome she believes more likely. In our example, the second expert will report her true beliefs (*$p_2^{(t)} = 1$*) while the first will not (*$p_1^{(t)} = 0$*). While the combined true beliefs of the experts clearly favor outcome* 1*, the WM algorithm sees two opposing predictions and must break ties arbitrarily between them.*

[7] The loss function is often tied to the particular application. For example, in the current FiveThirtyEight pollster rankings, the performance of a pollster is primarily measured according to an absolute loss function and also whether the candidate with the highest polling numbers ended up winning (see `https://github.com/fivethirtyeight/data/tree/master/pollster-ratings`). However, in 2008 FiveThirtyEight used the notion of "pollster introduced error" or PIE, which is the square root of a difference of squares, as the most important feature in calculating the weights, see `https://fivethirtyeight.com/features/pollster-ratings-v31/`.

This conclusion can also be drawn directly from the property elicitation literature. Here, the absolute loss function is known to elicit the median [Bonin, 1976][Thomson, 1979], and since we have binary realizations, the median is either 0 or 1. Example 5.1 shows that for the absolute loss function the standard WM algorithm is not "incentive-compatible" in a sense that we formalize in Section 5.2. There are similar examples for the other commonly studied weight update rules and for the RWM algorithm. We might care about truthful reporting for its own sake, but additionally the worry is that non-truthful reports will impede our ability to get good regret guarantees (with respect to experts' true losses).

We study several fundamental questions about online prediction with selfish experts:

1. What is the design space of "incentive-compatible" online prediction algorithms, where every expert is incentivized to report her true beliefs?

2. Given a loss function like absolute loss, are there incentive-compatible algorithm that obtain good regret guarantees?

3. Is online prediction with selfish experts strictly harder than in the classical model with honest experts?

### 5.1.1 Our Results

The first contribution of this paper is the development of a model for reasoning formally about the design and analysis of weight-based online prediction algorithms when experts are selfish (Section 5.2), and the definition of an "incentive-compatible" (IC) such algorithm. Intuitively, an IC algorithm is such that each expert wants to report its true belief at each time step. We demonstrate that the design of IC online prediction algorithms is closely related to the design of strictly proper scoring rules. Using this, we show that for the quadratic loss function, the standard WM and RWM algorithms are IC, whereas these algorithms are not generally IC for other loss functions.

Our second contribution is the design of IC prediction algorithms for the absolute loss function with non-trivial performance guarantees. For example, our best result for deterministic algorithms is: the WM algorithm, with experts' weights evolving according to the spherical proper scoring rule (see Section 5.3), is IC and has loss at most $2 + \sqrt{2}$ times the loss of best expert in hindsight (in the limit as $T \to \infty$). A variant of the RWM algorithm with the Brier scoring rule is IC and has expected loss at most 2.62 times that of the best expert in hindsight (also in the limit, see Section 5.6).

Our third and most technical contribution is a formal separation between online prediction with selfish experts and the traditional setting with honest experts. Recall

that with honest experts, the classical (deterministic) WM algorithm has loss at most twice that of the best expert in hindsight (as $T \to \infty$) Littlestone and Warmuth [1994]. We prove that the worst-case loss of every (deterministic) IC algorithm (Section 5.4) and every non-IC algorithm satisfying mild technical conditions (Section 5.5) has worst-case loss bounded away from twice that of the best expert in hindsight (even as $T \to \infty$). A consequence of our lower bound is that, with selfish experts, there is no natural (randomized) algorithm for online prediction—IC or otherwise—with asymptotically vanishing regret.

### 5.1.2 Related Work

We believe that our model of online prediction over time with selfish experts is novel. We next survey the multiple other ways in which online learning and incentive issues have been blended, and the other efforts to model incentive issues in machine learning.

There is a large literature on prediction and decision markets (e.g. [Chen and Pennock, 2010, Horn et al., 2014]), which also aim to aggregate information over time from multiple parties and make use of proper scoring rules to do it. There are several major differences between our model and prediction markets. First, in our model, the goal is to predict a sequence of events, and there is feedback (i.e., the realization) after each one. In a prediction market, the goal is to aggregate information about a single event, with feedback provided only at the end (subject to secondary objectives, like bounded loss).[8] Second, our goal is to make accurate predictions, while that of a prediction market is to aggregate information. For example, if one expert is consistently incorrect over time, we would like to ignore her reports rather than aggregate them with others' reports. Finally, while there are strong mathematical connections between cost function-based prediction markets and regularization-based online learning algorithms in the standard (non-IC) model [Abernethy et al., 2013], there does not appear to be analogous connections with online prediction with selfish experts.

There is also an emerging literature on "incentivizing exploration" (as opposed to exploitation) in partial feedback models such as the bandit model (e.g. [Frazier et al., 2014, Mansour et al., 2016]). Here, the incentive issues concern the learning algorithm itself, rather than the experts (or "arms") that it makes use of.

The question of how an expert should report beliefs has been studied before in the literature on strictly proper scoring rules [Brier, 1950, McCarthy, 1956, Savage, 1971, Gneiting and Raftery, 2007], but this literature typically considers the evaluation of a single prediction, rather than low-regret learning. The work by Bayarri and DeGroot [1989] specifically looks at the question of how an expert should respond

[8] In the even more distantly related peer prediction scenario [Miller et al., 2005], there is never any realization at all.

to an aggregator who assigns and updates weights based on their predictions. Their work focuses on optimizing relative weight under different objectives and informational assumptions. However, it predates the work on low-regret learning, and it does not include performance guarantees for the aggregator over time. Boutilier [2012] discusses a model in which an aggregator wants to take a specific action based on predictions that she elicits from experts. He explores incentive issues where experts have a stake in the action that is taken by the decision maker.

Finally, there are many works that fall under the broader umbrella of incentives in machine learning. Roughly, work in this area can be divided into two genres: incentives during the learning stage, or incentives during the deployment stage. During the learning stage, one of the main considerations is incentivizing data providers to exert effort to generate high-quality data. There are several recent works that propose ways to elicit data in crowdsourcing applications in repeated settings through payments, e.g. [Cai et al., 2015, Shah and Zhou, 2015, Liu and Chen, 2016]. Outside of crowdsourcing, Dekel et al. [2010] consider a regression task where different experts have their own private data set, and they seek to influence the learner to learn a function such that the loss of their private data set with respect to the function is low.

During deployment, the concern is that the input is given by agents who have a stake in the result of the classification, e.g. an email spammer wishes to avoid its emails being classified as spam. Brückner and Scheffer [2011] model a learning task as a Stackelberg game. On the other hand Hardt et al. [2016] consider a cost to changing data, e.g. improving your credit score by opening more lines of credit, and give results with respect to different cost functions.

Online learning does not fall neatly into either learning or deployment, as the learning is happening while the system is deployed. Babaioff et al. [2010] consider the problem of no-regret learning with selfish experts in an ad auction setting, where the incentives come from the allocations and payments of the auction, rather than from weights as in our case.

### 5.1.3 Organization

Section 5.2 formally defines weight-update online prediction algorithms and shows a connection between algorithms that are incentive compatible and proper scoring rules. We use the formalization to show that when we care about achieving guarantees for quadratic losses, the standard WM and RWM algorithms work well. Since the standard algorithm fails to work well for absolute losses, we focus in the remainder of the paper on proving guarantees for this case.

Section 5.3 gives a deterministic weight-update online prediction algorithm that is incentive-compatible and has absolute loss at most $2 + \sqrt{2}$ times that of the best expert in hindsight (in the limit). Additionally we show that the weighted majority

algorithm with the standard update rule has a worst-case true loss of at least 4 times the best expert in hindsight.

To show the limitations of online prediction with selfish experts, we break our lower bound results into two parts. In Section 5.4 we show that any deterministic incentive compatible weight-update online prediction algorithm has worst case loss bounded away from 2, even as $T \to \infty$. Then in Section 5.5 we show that under mild technical conditions, the same is true for non-IC algorithms.

Section 5.6 contains our results for randomized algorithms. It shows that the lower bounds for deterministic algorithms imply that under the same conditions randomized algorithms cannot have asymptotically vanishing regret. We do give an IC randomized algorithm that achieves worst-case loss at most 2.62 times that of the best expert in hindsight (in the limit).

Finally, in Section 5.7 we show simulations that indicate that different IC methods show similar regret behavior, and that their regret is substantially better than that of the non-IC standard algorithms, suggesting that the worst-case characterization we prove holds more generally.

The appendix contains omitted proofs (Appendix 5.A), and a discussion on the selecting appropriate proper scoring rules for good guarantees (Appendix 5.B).

## 5.2 Preliminaries and Model

### 5.2.1 Standard Model

At each time step $t \in 1, ..., T$ we want to predict a binary realization $r^{(t)} \in \{0, 1\}$. To help in the prediction, we have access to $n$ experts that for each time step report a prediction $p_i^{(t)} \in [0, 1]$ about the realization. The realizations are determined by an oblivious adversary, and the predictions of the experts may or may not be accurate. The goal is to use the predictions of the experts in such a way that the algorithm performs nearly as well as the best expert in hindsight. Most of the algorithms proposed for this problem fall into the following framework.

**Definition 5.2** (Weight-update Online Prediction Algorithm)**.** *A weight-update online prediction algorithm maintains a weight $w_i^{(t)}$ for each expert and makes its prediction $q^{(t)}$ based on $\sum_{i=1}^{n} w_i^{(t)} p_i^{(t)}$ and $\sum_i^n w_i^{(t)}(1 - p_i^{(t)})$. After the algorithm makes its prediction, the realization $r^{(t)}$ is revealed, and the algorithm updates the weights of experts using the rule*

$$w_i^{(t+1)} = f\left(p_i^{(t)}, r^{(t)}\right) \cdot w_i^{(t)}, \tag{5.1}$$

*where $f : [0, 1] \times \{0, 1\} \to \mathbb{R}^+$ is a positive function on its domain.*

The standard WM algorithm has $f(p_i^{(t)}, r^{(t)}) = 1 - \eta\ell(p_i^{(t)}, r^{(t)})$ where $\eta \in (0, \frac{1}{2})$ is the learning rate, and predicts $q^{(t)} = 1$ if and only if $\sum_i^n w_i^{(t)} p_i^{(t)} \geq \sum_i^n w_i^{(t)}(1 - p_i^{(t)})$. Let the total loss of the algorithm be $M^{(T)} = \sum_{t=1}^T \ell(q^{(t)}, r^{(t)})$ and let the total loss of expert $i$ be $m_i^{(T)} = \sum_{t=1}^T \ell(p_i^{(t)}, r^{(t)})$. The MW algorithm has the property that $M^{(T)} \leq 2(1+\eta)m_i^{(T)} + \frac{2\ln n}{\eta}$ for each expert $i$, and RWM —where the algorithm picks 1 with probability proportional to $\sum_i^n w_i^{(t)} p_i^{(t)}$— satisfies $M^{(T)} \leq (1+\eta)m_i^{(T)} + \frac{\ln n}{\eta}$ for each expert $i$ [Littlestone and Warmuth, 1994][Freund and Schapire, 1997].

The notion of *"no $\alpha$-regret"* [Kakade et al., 2009] captures the idea that the per time-step loss of an algorithm is $\alpha$ times that of the best expert in hindsight, plus a term that goes to 0 as $T$ grows:

**Definition 5.3** ($\alpha$-regret)**.** *An algorithm is said to have* no $\alpha$-regret *if* $M^{(T)} \leq \alpha \min_i m_i^{(T)} + o(T)$.

By taking $\eta = O(1/\sqrt{T})$, MW is a no 2-regret algorithm, and RWM is a no 1-regret algorithm.

### 5.2.2 Selfish Model

We consider a model in which experts have agency about the prediction they report, and care about the weight that they are assigned. In the selfish model, at time $t$ the expert formulates a private belief $b_i^{(t)}$ about the realization, but she is free to report any prediction $p_i^{(t)}$ to the algorithm. Let Bern($p$) be a Bernoulli random variable with parameter $p$. For any non-negative weight update function $f$,

$$\max_p \mathbb{E}_{b_i^{(t)}}[w_i^{(t+1)}] \;=\; \max_p \mathbb{E}_{r\sim\text{Bern}\left(b_i^{(t)}\right)}[f\,(p,r)\,w_i^{(t)}] \;=\; w_i^{(t)} \cdot \left(\max_p \mathbb{E}_{r\sim\text{Bern}\left(b_i^{(t)}\right)}[f\,(p,r)]\right).$$

So expert $i$ will report whichever $p_i^{(t)}$ will maximize the expectation of the weight update function.

Performance of an algorithm with respect to the reported loss of experts follows from the standard analysis [Littlestone and Warmuth, 1994]. However, the true loss may be worse (in Section 5.3 we show this for the standard update rule, Section 5.5 shows it more generally). Unless explicitly stated otherwise, in the remainder of this paper $m_i^{(T)} = \sum_{t=1}^T \ell(b_i^{(t)}, r^{(t)})$ refers to the *true* loss of expert $i$. For now this motivates restricting the weight update rule $f$ to functions where reporting $p_i^{(t)} = b_i^{(t)}$ maximizes the expected weight of experts. We call these weight-update rules *Incentive Compatible (IC)*.

**Definition 5.4** (Incentive Compatibility)**.** *A weight-update function $f$ is* incentive compatible (IC) *if reporting the true belief $b_i^{(t)}$ is always a best response for every expert at every time step. It is* strictly IC *when $p_i^{(t)} = b_i^{(t)}$ is the only best response.*

By a "best response," we mean an expected utility-maximizing report, where the expectation is with respect to the expert's beliefs.

**Collusion.** The definition of IC does not rule out the possibility that experts can collude to jointly misreport to improve their weights. We therefore also consider a stronger notion of incentive compatibility for groups with transferable utility.

**Definition 5.5** (Incentive Compatibility for Groups with Transferable Utility)**.** *A weight-update function $f$ is* incentive compatible for groups with transferable utility (TU-GIC) *if for every subset $S$ of players, the total expected weight of the group $\sum_{i \in S} \mathbb{E}_{b_i^{(t)}}[w_i^{(t+1)}]$ is maximized by each reporting their private belief $b_i^{(t)}$.*

Note that TU-GIC is a strictly stronger concept than IC, as for any algorithm that is TU-GIC, the condition needs to hold for groups of size 1, which is the definition of IC. The concept is also strictly stronger than that of GIC with nontransferable utility (NTU-GIC), where for every group $S$ it only needs to hold that there are no alternative reports that would make no member worse off, and at least one member better off [Moulin, 1999][Jain and Mahdian, 2007].

### 5.2.3 Proper Scoring Rules

Incentivizing truthful reporting of beliefs has been studied extensively, and the set of functions that do this is called the set of proper scoring rules. Since we focus on predicting a binary event, we restrict our attention to this class of functions.

**Definition 5.6** (Binary Proper Scoring Rule, [Schervish, 1989])**.** *A function $f : [0,1] \times \{0,1\} \to \mathbb{R} \cup \{\pm\infty\}$ is a* binary proper scoring rule *if it is finite except possibly on its boundary and whenever for $p \in [0,1]$*

$$p \in \max_{q \in [0,1]} p \cdot f(q,1) + (1-p) \cdot f(q,0).$$

A function $f$ is a *strictly* proper scoring rule if $p$ is the only value that maximizes the expectation. The first perhaps most well-known proper scoring rule is the Brier scoring rule.

**Example 5.7** (Brier Scoring Rule, [Brier, 1950])**.** *The* Brier score *is $Br(p,r) = 2p_r - (p^2 + (1-p)^2)$ where $p_r = pr + (1-p)(1-r)$ is the report for the event that materialized.*

We will use the Brier scoring rule in Section 5.6 to construct an incentive-compatible randomized algorithm with good guarantees. The following proposition follows directly from Definitions 5.4 and 5.6:

**Proposition 5.8.** *A weight-update rule $f$ is (strictly) incentive compatible if and only if $f$ is a (strictly) proper scoring rule.*

Surprisingly, this result remains true even when experts can collude.While the realizations are obviously correlated, linearity of expectation causes the sum to be maximized exactly when each expert maximizes their expected weight.

**Proposition 5.9.** *A weight-update rule $f$ is (strictly) incentive compatible for groups with transferable utility if and only if $f$ is a (strictly) proper scoring rule.*

Thus, for online prediction with selfish experts, we get TU-GIC "for free."It is quite uncommon for problems in non-cooperate game theory to admit good TU-GIC solutions. For example, results for auctions (either for revenue or welfare) break down once bidders collude, see [Goldberg and Hartline, 2005] and references therein for more examples from theory and practice. In the remainder of the paper we will simply us IC to refer to both incentive compatibility and incentive compatibility for groups with transferable utility, as strictly proper scoring rules lead to algorithms that satisfy both definitions.

So when considering incentive compatibility in the online prediction with selfish experts setting, we are restricted to considering proper scoring rules as weight-update rules. Moreover, since $f$ needs to be positive, only bounded proper scoring rules can be used. Conversely, any bounded scoring rule can be used, possibly after an affine transformation (which preserves proper-ness). Are there any proper scoring rules that give an online prediction algorithm with a good performance guarantee?

### 5.2.4 Online Learning with Quadratic Losses

The first goal of this paper is to describe the class of algorithms that lead incentive compatible learning. Proposition 5.8 answers this question, so we can move over to our second goal, which is for different loss functions, do there exist incentive compatible algorithms with good performance guarantees? In this subsection we show that a corollary of Proposition 5.8 is that the standard MW algorithm with the quadratic loss function $\ell(p, r) = (p - r)^2$ is incentive compatible.

**Corollary 5.10.** *The standard WM algorithm with quadratic losses, i.e. $w_i^{(t+1)} = (1 - \eta(p_i^{(t)} - r^{(t)}))^2 \cdot w_i^{(t)}$ is incentive compatible.*

*Proof.* By Proposition 5.8 it is sufficient to show that $b_i^{(t)} = \max_p b_i^{(t)} \cdot (1 - \eta(p-1)^2) + (1 - b_i^{(t)}) \cdot (p-0)^2$.

$$\begin{aligned}
&\max_p b_i^{(t)} \cdot (1 - \eta(p-1)^2) + (1 - b_i^{(t)}) \cdot (1 - \eta(p-0)^2) \\
=&\max_p b_i^{(t)} - b_i^{(t)}\eta p^2 + 2b_i^{(t)}\eta p - b_i^{(t)}\eta + 1 - b_i^{(t)} - \eta p^2 + b_i^{(t)}\eta p^2 \\
=&\max_p 1 - b_i^{(t)}\eta + 2b_i^{(t)}\eta p - \eta p^2 \\
=&\max_p 1 - b_i^{(t)}\eta + \eta p(2b_i^{(t)} - p)
\end{aligned}$$

To solve this for $p$, we take the derivative with respect to $p$: $\frac{d}{dp} 1 - b_i^{(t)}\eta + \eta p(2b_i^{(t)} - p) = \eta(2b_i^{(t)} - 2p)$. So the maximum expected value is uniquely $p = b_i^{(t)}$. □

A different way of proving the Corollary is by showing that the standard update rule with quadratic losses can be translated into the Brier strictly proper scoring rule. Either way, for applications with quadratic losses, the standard algorithm already works out of the box. However, as we saw in Example 5.1, this is not the case with the absolute loss function. As the absolute loss function arises in practice—recall that FiveThirtyEight uses absolute loss to calculate their pollster ratings—in the remainder of this paper we focus on answering questions (2) and (3) from the introduction for the absolute loss function.

## 5.3 Deterministic Algorithms for Selfish Experts

This section studies the question if there are good online prediction algorithms for the absolute loss function. We restrict our attention here to deterministic algorithms; Section 5.6 gives a randomized algorithm with good guarantees.

Proposition 5.8 tells us that for selfish experts to have a strict incentive to report truthfully, the weight-update rule must be a strictly proper scoring rule. This section gives a deterministic algorithm based on the *spherical* strictly proper scoring rule that has no $(2+\sqrt{2})$-regret (Theorem 5.12). Additionally, we consider the question if the non-truthful reports from experts in using the standard (non-IC) WM algorithm are harmful. We show that this is the case by proving that the algorithm is not a no $(4 - O(1))$-regret algorithm, for any constant smaller than 4 (Proposition 5.13). This shows that, when experts are selfish, the IC online prediction algorithm with the spherical rule outperforms the standard WM algorithm (in the worst case).

### 5.3.1 Deterministic Online Prediction using a Spherical Rule

We next give an algorithm that uses a strictly proper scoring rule that is based on the spherical rule scoring rule.[9] In the following, let $s_i^{(t)} = |p_i^{(t)} - r^{(t)}|$ be the absolute loss of expert $i$.

Consider the following weight-update rule:

$$f_{sp}\left(p_i^{(t)}, r^{(t)}\right) = 1 - \eta\left(1 - \frac{1 - s_i^{(t)}}{\sqrt{\left(p_i^{(t)}\right)^2 + \left(1 - p_i^{(t)}\right)^2}}\right). \tag{5.2}$$

The following proposition establishes that this is in fact a strictly proper scoring rule.

**Proposition 5.11.** *The spherical weight-update rule in* (5.2) *is a strictly proper scoring rule.*

*Proof.* The standard spherical strictly proper scoring rule is

$$\frac{1 - s_i^{(t)}}{\sqrt{(p_i^{(t)})^2 + (1 - p_i^{(t)})^2}}.$$

Any positive affine transformation of a strictly proper scoring rule yields another strictly proper scoring rule, see e.g. [Gneiting and Raftery, 2007], hence

$$\frac{1 - s_i^{(t)}}{\sqrt{(p_i^{(t)})^2 + (1 - p_i^{(t)})^2}} - 1$$

is also a strictly proper scoring rule. Now we multiply this by $\eta$ and add 1 to obtain

$$1 + \eta\left(\frac{1 - s_i^{(t)}}{\sqrt{\left(p_i^{(t)}\right)^2 + \left(1 - p_i^{(t)}\right)^2}} - 1\right),$$

and rewriting proves the claim. □

In addition to incentivizing truthful reporting, the WM algorithm with the update rule $f_{\text{sp}}$ does not do much worse than the best expert in hindsight. (See the chapter appendix for the proof.)

[9] See Appendix 5.B for intuition about why this rule yields better results than other natural candidates, such as the Brier scoring rule.

**Theorem 5.12.** *The WM algorithm with weight-update rule* (5.2) *for* $\eta = O(1/\sqrt{T}) < \frac{1}{2}$ *has no* $(2+\sqrt{2})$*-regret.*

### 5.3.2 True Loss of the Non-IC Standard Rule

It is instructive to compare the guarantee in Theorem 5.12 with the performance of the standard (non-IC) WM algorithm. With the standard weight update function $f(p_i^{(t)}, r^{(t)}) = 1 - \eta s_i^{(t)}$ for $\eta \in (0, \frac{1}{2})$, the WM algorithm has the guarantee that $M^{(T)} \leq 2\left((1+\eta)m_i^{(T)} + \frac{\ln n}{\eta}\right)$ with respect to the *reported* loss of experts. However, Example 5.1 demonstrates that this algorithm incentivizes extremal reports, i.e. if $b_i^{(t)} \in [0, \frac{1}{2})$ the expert will report $p_i^{(t)} = 0$ and if $b_i^{(t)} \in (\frac{1}{2}, 1]$ the expert will report 1. The following proposition shows that, in the worst case, this algorithm does no better than a factor 4 times the *true* loss of the best expert in hindsight. Theorem 5.12 shows that a suitable IC algorithm can obtain a superior worst-case guarantee.

**Proposition 5.13.** *The standard WM algorithm with weight-update rule* $f\left(p_i^{(t)}, r^{(t)}\right) = 1 - \eta|p_i^{(t)} - r^{(t)}|$ *results in a total worst-case loss no better than*

$$M^{(T)} \geq 4 \cdot \min_i m_i^{(T)} - o(1).$$

*Proof.* Let $A$ be the standard weighted majority algorithm. We create an instance with 2 experts where $M^{(T)} \geq 4 \cdot \min_i m_i^{(T)} - o(1)$. Let the reports $p_1^{(t)} = 0$, and $p_2^{(t)} = 1$ for all $t \in 1, ..., T$; we will define $b_i^{(t)}$ shortly. Given the reports, $A$ will choose a sequence of predictions, let $r^{(t)}$ be 1 whenever the algorithm chooses 0 and vice versa, so that $M^{(T)} = T$.

Now for all $t$ such that $r^{(t)} = 1$, set $b_1^{(t)} = \frac{1}{2} - \epsilon$ and $b_2^{(t)} = 1$, and for all $t$ such that $r^{(t)} = 0$ set $b_1^{(t)} = 0$ and $b_2^{(t)} = \frac{1}{2} + \epsilon$, for small $\epsilon > 0$. Note that the beliefs $b_i^{(t)}$ indeed lead to the reports $p_i^{(t)}$ since $A$ incentivizes rounding the reports to the nearest integer.

Since the experts reported opposite outcomes, their combined total number of incorrect reports is $T$, hence the best expert had a reported loss of at most $T/2$. For each incorrect report $p_i^{(t)}$, the real loss of expert is $|r^{(t)} - b_i^{(t)}| = \frac{1}{2} + \epsilon$, hence $\min_i m_i^{(T)} \leq \left(\frac{1}{2} + \epsilon\right) T/2$, while $M^{(T)} = T$. Taking $\epsilon = o(T^{-1})$ yields the claim. □

## 5.4 The Cost of Selfish Experts for IC algorithms

We now address the third fundamental question: whether or not online prediction with selfish experts is strictly harder than with honest experts. In this section we

restrict our attention to deterministic algorithms; we extend the results to randomized algorithms in Section 5.6. As there exists a deterministic algorithm for honest experts with no 2-regret, showing a separation between honest and selfish experts boils down to proving that there exists a constant $\delta$ such that the worst-case loss is no better than a factor of $2 + \delta$ (with $\delta$ bounded away from 0 as $T \to \infty$).

In this section we show that such a $\delta$ exists for all incentive compatible algorithms, and that $\delta$ depends on properties of a "normalized" version of the weight-update rule $f$, independent of the learning rate. This implies that the lower bound also holds for algorithms that, like the classical prediction algorithms, use a time-varying learning rate. In Section 5.5 we show that under mild technical conditions the true loss of non-IC algorithms is also bounded away from 2, and in Section 5.6 the lower bounds for deterministic algorithms are used to show that there is no randomized algorithm that achieves vanishing regret.

To prove the lower bound, we have to be specific about which set of algorithms we consider. To cover algorithms that have a decreasing learning parameter, we first show that any positive proper scoring rule can be interpreted as having a learning parameter $\eta$.

**Proposition 5.14.** *Let $f$ be any strictly proper scoring rule. We can write $f$ as $f(p, r) = a + bf'(p, r)$ with $a \in \mathbb{R}$, $b \in \mathbb{R}^+$ and $f'$ a strictly proper scoring rule with $\min(f'(0, 1), f'(1, 0)) = 0$ and $\max(f'(0, 0), f'(1, 1)) = 1$.*

*Proof.* Let $f_{min} = \min(f(0, 1), f(1, 0))$ and $f_{max} = \max(f(0, 0), f(1, 1)) = 1$. Then define $f'(p, r) = \frac{f(p,r)-f_{min}}{f_{max}-f_{min}}$, $a = f_{min}$ and $b = f_{max} - f_{min}$. This is a positive affine translation, hence $f'$ is a strictly proper scoring rule. □

We call $f' : [0, 1] \times \{0, 1\} \to [0, 1]$ a *normalized* scoring rule. Using normalized scoring rules, we can define a family of scoring rules with different learning rates $\eta$.

**Definition 5.15.** *Let $f$ be any normalized strictly proper scoring rule. Define $\mathcal{F}$ as the following family of proper scoring rules generated by $f$:*

$$\mathcal{F} = \{f'(p, r) = a\,(1 + \eta(f(p, r) - 1)) : a > 0 \text{ and } \eta \in (0, 1)\}$$

By Proposition 5.14 the union of families generated by normalized strictly proper scoring rules cover all strictly proper scoring rules. Using this we can now formulate the class of deterministic algorithms that are incentive compatible.

**Definition 5.16** (Deterministic Incentive-Compatible Algorithms)**.** *Let $\mathcal{A}_d$ be the set of deterministic algorithms that update weights by $w_i^{(t+1)} = a(1 + \eta(f(p_i^{(t)}, r^{(t)}) - 1))w_i^{(t)}$, for a normalized strictly proper scoring rule $f$ and $\eta \in (0, \frac{1}{2})$ with $\eta$ possibly decreasing over time. For $q = \sum_{i=1}^n w_i^{(t)} p_i^{(t)} / \sum_{i=1}^n w_i^{(t)}$, $A$ picks $q^{(t)} = 0$ if $q < \frac{1}{2}$, $q^{(t)} = 1$ if $q > \frac{1}{2}$ and uses any deterministic tie breaking rule for $q = \frac{1}{2}$.*

Using this definition we can now state our main result:

**Theorem 5.17.** *For the absolute loss function, there does not exists a deterministic and incentive-compatible algorithm $A \in \mathcal{A}_d$ with no 2-regret.*

To prove Theorem 5.17 we proceed in two steps. First we consider strictly proper scoring rules that are symmetric with respect to the outcomes, because they lead to a lower bound that can be naturally interpreted by looking at the geometry of the scoring rule. We then extend these results to cover weight-update rules that use any (potentially asymmetric) strictly proper scoring rule.

### 5.4.1 Symmetric Strictly Proper Scoring Rules

We first focus on symmetric scoring rules in the sense that $f(p, 0) = f(1 - p, 1)$ for all $p \in [0, 1]$. We can thus write these as $f(p) = f(p, 1) = f(1 - p, 0)$. Many common scoring rules are symmetric, including the Brier rule [Brier, 1950], the family of pseudo-spherical rules (e.g. [Gneiting and Raftery, 2007]), the power family (e.g. [Jose et al., 2008]), and the beta family [Buja et al., 2005] when $\alpha = \beta$. We start by defining the scoring rule gap for normalized scoring rules, which will determine the lower bound constant.

**Definition 5.18** (Scoring Rule Gap)**.** *The* scoring rule gap $\gamma$ *of family $\mathcal{F}$ with generator $f$ is $\gamma = f(\frac{1}{2}) - \frac{1}{2}(f(0) + f(1)) = f(\frac{1}{2}) - \frac{1}{2}$.*

The following proposition shows that for all strictly proper scoring rules, the scoring rule gap must be strictly positive.

**Proposition 5.19.** *The scoring rule gap $\gamma$ of a family generated by a symmetric strictly proper scoring rule $f$ is strictly positive.*

*Proof.* Since $f$ is symmetric and a strictly proper scoring rule, we must have that $\frac{1}{2}f(\frac{1}{2}) + \frac{1}{2}f(\frac{1}{2}) > \frac{1}{2}f(0) + \frac{1}{2}f(1)$ (since an expert with belief $\frac{1}{2}$ must have a strict incentive to report $\frac{1}{2}$ instead of 1). The statement follows from rewriting. □

We are now ready to prove our lower bound for all symmetric strictly proper scoring rules. The interesting case is where the learning rate $\eta \to 0$, as otherwise it is easy to prove a lower bound bounded away from 2.

The following lemma establishes that the gap parameter is important in proving lower bounds for IC online prediction algorithms. Intuitively, the lower bound instance exploits that experts who report $\frac{1}{2}$ will have a higher weight (due to the scoring rule gap) than an expert who is alternatingly right and wrong with extreme reports. This means that even though the indifferent expert has the same absolute loss, she will have a higher weight and this can lead the algorithm astray. The scoring rule gap is also relevant for the discussion in Appendix 5.B. We give partial proof of the lemma below; the full proof appears in Appendix 5.A.

**Lemma 5.20.** *Let $\mathcal{F}$ be a family of scoring rules generated by a symmetric strictly proper scoring rule $f$, and let $\gamma$ be the scoring rule gap of $\mathcal{F}$. In the worst case, MW with any scoring rule $f' \in \mathcal{F}$ with $\eta \in (0, \frac{1}{2})$, algorithm loss $M^{(T)}$ and expert loss $m_i^{(T)}$, satisfies*

$$M^{(T)} \geq \left(2 + \frac{1}{\lceil \gamma^{-1} \rceil}\right) \cdot m_i^{(T)}.$$

*Proof Sketch.* Let $a$, $\eta$ be the parameters of $f'$ in the family $\mathcal{F}$, as in Definition 5.15. Fix $T$ sufficiently large and an integer multiple of $2\lceil \gamma^{-1} \rceil + 1$, and let $e_1$, $e_2$, and $e_3$ be three experts. For $t = 1, ..., \alpha \cdot T$ where $\alpha = \frac{2\lceil \gamma^{-1} \rceil}{2\lceil \gamma^{-1} \rceil + 1}$ such that $\alpha T$ is an even integer, let $p_1^{(t)} = \frac{1}{2}$, $p_2^{(t)} = 0$, and $p_3^{(t)} = 1$. Fix any tie-breaking rule for the algorithm. Realization $r^{(t)}$ is always the opposite of what the algorithm chooses.

Let $M^{(t)}$ be the loss of the algorithm up to time $t$, and let $m_i^{(t)}$ be the loss of expert $i$. We first show that at $t' = \alpha T$, $m_1^{(t')} = m_2^{(t')} = m_3^{(t')} = \frac{\alpha T}{2}$ and $M^{(t')} = \alpha T$. The latter part is obvious as $r^{(t)}$ is the opposite of what the algorithm chooses. That $m_1^{(t')} = \frac{\alpha T}{2}$ is also obvious as it adds a loss of $\frac{1}{2}$ at each time step. To show that $m_2^{(t')} = m_3^{(t')} = \frac{\alpha T}{2}$ we do induction on the number of time steps, in steps of two. The induction hypothesis is that after an even number of time steps, $m_2^{(t)} = m_3^{(t)}$ and that $w_2^{(t)} = w_3^{(t)}$. Initially, all weights are 1 and both experts have loss of 0, so the base case holds. Consider the algorithm after an even number $t$ time steps. Since $w_2^{(t)} = w_3^{(t)}$, $p_3^{(t)} = 1 - p_2^{(t)}$, and $p_1^{(t)} = \frac{1}{2}$ we have that $\sum_{i=1}^3 w_i^{(t)} p_i^{(t)} = \sum_{i=1}^3 w_i^{(t)}(1 - p_i^{(t)})$ and hence the algorithm will use its tie-breaking rule for its next decision. Thus, either $e_2$ or $e_3$ is wrong. Wlog let's say that $e_2$ was wrong (the other case is symmetric), so $m_2^{(t+1)} = 1 + m_3^{(t+1)}$. Now at time $t+1$, $w_2^{(t+1)} = (1-\eta) w_3^{(t+1)} < w_3^{(t+1)}$. Since $e_1$ does not express a preference, and $e_3$ has a higher weight than $e_2$, the algorithm will follow $e_3$'s advice. Since the realization $r^{(t+1)}$ is the opposite of the algorithms choice, this means that now $e_3$ incurs a loss of one. Thus $m_2^{(t+2)} = m_2^{(t+1)}$ and $w_2^{(t+2)} = w_2^{(t+1)}$ and $m_3^{(t+2)} = 1 + m_3^{(t+1)} = m_2^{(t+2)}$. The weight of expert $e_2$ is $w_2^{(t+2)} = aa(1-\eta) w_2^{(t)}$ and the weight of expert $e_3$ is $w_3^{(t+2)} = a(1-\eta) a w_3^{(t)}$. By the induction hypothesis $w_2^{(t)} = w_3^{(t)}$, hence $w_2^{(t+2)} = w_3^{(t+2)}$, and since we already showed that $m_2^{(t+2)} = m_3^{(t+2)}$, this completes the induction.

Now, for $t = \alpha T + 1, ..., T$, we let $p_1^{(t)} = 1$, $p_2^{(t)} = 0$, $p_3^{(t)} = \frac{1}{2}$ and $r^{(t)} = 0$. So henceforth $e_3$ does not provide information, $e_1$ is always wrong, and $e_2$ is always right. If we can show that the algorithm will always follow $e_1$, then the best expert is $e_2$ with a loss of $m_2^{(T)} = \frac{\alpha T}{2}$, while the algorithm has a loss of $M^{(T)} = T$. If this holds for $\alpha < 1$ this proves the claim. So what's left to prove is that the algorithm will always follow $e_1$. Note that since $p_3^{(t)} = \frac{1}{2}$ it contributes equal amounts to $\sum_{i=1}^3 w_i^{(t)} p_i^{(t)}$ and $\sum_{i=1}^3 w_i^{(t)}(1 - p_i^{(t)})$ and is therefore ignored by the algorithm in making its decision. So it suffices to look at $e_1$ and $e_2$. The algorithm will pick 1 iff

$\sum_{i=1}^{3} w_i^{(t)}(1-p_i^{(t)}) \leq \sum_{i=1}^{3} w_i^{(t)} p_i^{(t)}$, which after simplifying becomes $w_1^{(t)} > w_2^{(t)}$.

At time step $t$, $w_1^{(t)} = \big(a(1+\eta(f(\frac{1}{2})-1))\big)^{\alpha T}\,(a\cdot(1-\eta))^{t-\alpha T}$ for expert $e_1$, and $w_2^{(t)} = (a(1-\eta))^{\frac{\alpha T}{2}}\, a^{\frac{\alpha T}{2}+t-\alpha T}$ for $e_2$. We have that $w_1^{(t)}$ is decreasing faster in $t$ than $w_2^{(t)}$. So if we can show that $w_1^{(T)} \geq w_2^{(T)}$ for some $\alpha < 1$, then $e_2$ will incur a total loss of $\alpha T/2$ while the algorithm incurs a loss of $T$ and the statement is proved. This is shown in the appendix. □

As a consequence of Lemma 5.20, we can calculate lower bounds for specific strictly proper scoring rules. For example, the spherical rule used in Section 5.3.1 is a symmetric strictly proper scoring rule with a gap parameter $\gamma = \frac{\sqrt{2}}{2} - \frac{1}{2}$, and hence $1/\lceil\gamma^{-1}\rceil = \frac{1}{5}$.

**Corollary 5.21.** *In the worst case, the deterministic algorithm based on the spherical rule in Section 5.3.1 has*

$$M^{(T)} \geq \big(2+\tfrac{1}{5}\big)\, m_i^{(T)}.$$

We revisit the scoring rule gap parameter again in Appendix 5.B when we discuss considerations for selecting different scoring rules.

### 5.4.2 Beyond Symmetric Strictly Proper Scoring Rules

We now extend the lower bound example to cover arbitrary strictly proper scoring rules. As in the previous subsection, we consider properties of normalized scoring rules to provide lower bounds that are independent of learning rate, but the properties in this subsection have a less natural interpretation.

For arbitrary strictly proper scoring rule $f'$, let $f$ be the corresponding normalized scoring rule, with parameters $a$ and $\eta$. Since $f$ is normalized, $\max\{f(0,0), f(1,1)\} = 1$ and $\min\{f(0,1), f(1,0)\} = 0$. We consider 2 cases, one in which $f(0,0) = f(1,1) = 1$ and $f(0,1) = f(1,0) = 0$ which is locally symmetric, and the case where at least one of those equalities does not hold.

**The semi-symmetric case.** If it is the case that $f$ has $f(0,0) = f(1,1) = 1$ and $f(0,1) = f(1,0) = 0$, then $f$ has enough symmetry to prove a variant of the lower bound instance discussed just before. Define the semi-symmetric scoring rule gap as follows.

**Definition 5.22** (Semi-symmetric Scoring Rule Gap)**.** *The* 'semi-symmetric' scoring rule gap $\mu$ *of family* $\mathcal{F}$ *with normalized generator* $f$ *is* $\mu = \frac{1}{2}\big(f(\frac{1}{2},0) + f(\frac{1}{2},1)\big) - \frac{1}{2}$.

Like the symmetric scoring rule gap, $\mu > 0$ by definition, as there needs to be a strict incentive to report $\frac{1}{2}$ for experts with $b_i^{(t)} = \frac{1}{2}$. Next, observe that since

$f(\frac{1}{2},0), f(\frac{1}{2},1) \in [0,1]$ and $f(\frac{1}{2},0)+f(\frac{1}{2},1) = 1+2\mu$, it must be that $f(\frac{1}{2},0)\cdot f(\frac{1}{2},1) \geq 2\mu$. Using this it follows that:

$$\begin{aligned}
&\big(1+\eta(f(\tfrac{1}{2},0)-1)\big)\big(1+\eta(f(\tfrac{1}{2},1)-1)\big)\\
&= 1+\eta\cdot\big(f(\tfrac{1}{2},0)+f(\tfrac{1}{2},1)-2\big)+\eta^2\big(f(\tfrac{1}{2},0)\cdot f(\tfrac{1}{2},1)-f(\tfrac{1}{2},0)-f(\tfrac{1}{2},1)+1\big)\\
&= 1+\eta\cdot(1+2\mu-2)+\eta^2\big(f(\tfrac{1}{2},0)\cdot f(\tfrac{1}{2},1)-2\mu\big)\\
&\geq 1-\eta(1-2\mu)+\eta^2(2\mu-2\mu)\\
&= 1-\eta+2\mu\eta \qquad (5.3)
\end{aligned}$$

Now this can be used in the same way as we proved the setting before:

**Lemma 5.23.** *Let $\mathcal{F}$ be a family of scoring rules generated by a normalized strictly proper scoring rule $f$, with $f(0,0) = f(1,1)$ and $f(0,1) = f(1,0)$. In the worst case, MW with any scoring rule $f'$ from $\mathcal{F}$ with $\eta \in (0,\frac{1}{2})$ can do no better than*

$$M^{(T)} \geq \left(2+\frac{1}{\lceil\mu^{-1}\rceil}\right)\cdot m_i^{(T)}.$$

*Proof Sketch.* Take the same instance as used in Lemma 5.20, with $\alpha = \frac{2\lceil\mu^{-1}\rceil}{2\lceil\mu^{-1}\rceil+1}$. The progression of the algorithm up to $t=\alpha T$ is identical in this case, as expert $e_1$ is indifferent between outcomes, and $f(0,0)=f(1,1)$ and $f(0,1)=f(1,0)$ for experts $e_2$ and $e_3$. What remains to be shown is that the weight of $e_1$ will be higher at time $T$. At time $T$ the weights of $e_1$ and $e_2$ are:

$$\begin{aligned}
a^{-T}w_1^{(T)} &= \big(1+\eta(f(\tfrac{1}{2},0)-1)\big)^{\frac{\alpha T}{2}}\big(1+\eta(f(\tfrac{1}{2},1)-1)\big)^{\frac{\alpha T}{2}}(1-\eta)^{(1-\alpha)T}\\
a^{-T}w_2^{(T)} &= (1-\eta)^{\frac{\alpha T}{2}}.
\end{aligned}$$

Similarly to the symmetric case, wee know that $w_1^{(T)} > w_2^{(T)}$ if we can show that

$$\big(1+\eta(f(\tfrac{1}{2},0)-1)\big)^{\lceil\mu^{-1}\rceil}\big(1+\eta(f(\tfrac{1}{2},1)-1)\big)^{\lceil\mu^{-1}\rceil}(1-\eta) > (1-\eta)^{\lceil\mu^{-1}\rceil}.$$

By (5.3), it suffices to show that $(1-\eta+2\mu\eta)^{\lceil\mu^{-1}\rceil}(1-\eta) > (1-\eta)^{\lceil\mu^{-1}\rceil}$, which holds by following the derivation in the proof of Lemma 5.20 given in the appendix, starting at (5.6). □

**The asymmetric case.** We finally consider the setting where the weight-update rule is not symmetric, nor is it symmetric evaluated only at the extreme reports. The lower bound that we show is based on the amount of asymmetry at these extreme points, and is parametrized as follows.

**Definition 5.24.** *Let $c > d$ be parameters of a normalized strictly proper scoring rule $f$, such that $c = 1 - \max\{f(0,1), f(1,0)\}$ and $d = 1 - \min\{f(0,0), f(1,1)\}$.*

Scoring rules that are not covered by Lemmas 5.20 or 5.23 must have either $c < 1$ or $d > 0$ or both. The intuition behind the lower bound instance is that two experts who have opposite predictions, and are alternatingly right and wrong, will end up with different weights, even though they have the same loss. We use this to show that eventually one expert will have a lower loss, but also a lower weight, so the algorithm will follow the other expert. This process can be repeated to get the bounds in the Lemma below. The proof of the lemma appears in the appendix.

**Lemma 5.25.** *Let $\mathcal{F}$ be a family of scoring rules generated by a normalized strictly proper scoring rule $f$, with not both $f(0,0) = f(1,1)$ and $f(0,1) = f(1,0)$ and parameters $c$ and $d$ as in Definition 5.24. In the worst case, MW with any scoring rule $f'$ from $\mathcal{F}$ with $\eta \in (0, \frac{1}{2})$ can do no better than*

$$M^{(T)} \geq \left(2 + \max\{\tfrac{1-c}{2c}, \tfrac{d}{4(1-d)}\}\right) \cdot m_i^{(T)}.$$

Theorem 5.17 now follows from combining the previous three lemmas.

*Proof of Theorem 5.17.* Follows from combining Lemmas 5.20, 5.23 and 5.25. □

## 5.5 The Cost of Selfish Experts for Non-IC Algorithms

What about non-incentive-compatible algorithms? Could it be that, even with experts reporting strategically instead of honestly, there is a deterministic no 2-regret algorithm (or a randomized algorithm with vanishing regret), to match the classical results for honest experts? Proposition 5.13 shows that the standard algorithm fails to achieve such a regret bound, but maybe some other non-IC algorithm does?

Typically, one would show that this is not the case by a "revelation principle" argument: if there exists some (non-IC) algorithm $A$ with good guarantees, then we can construct an algorithm $B$ which takes private values as input, and runs algorithm $A$ on whatever reports a self-interested agent would have provided to $A$. It does the strategic thinking for agents, and hence $B$ is an IC algorithm with the same performance as $A$. This means that generally, whatever performance is possible with non-IC algorithms can be achieved by IC algorithms as well, thus lower bounds for IC algorithms translate to lower bounds for non-IC algorithms. In our case however, the reports impact both the weights of experts as well as the decision of the algorithm simultaneously. Even if we insist on keeping the weights in $A$ and $B$ the same, the

decisions of the algorithms may still be different. Therefore, rather than relying on a simulation argument, we give a direct proof that, under mild technical conditions, non-IC deterministic algorithms cannot be no 2-regret.[10] As in the previous section, we focus on deterministic algorithms; Section 5.6 translates these lower bounds to randomized algorithms, where they imply that no vanishing-regret algorithms exist.

The following definition captures how players are incentivized to report differently from their beliefs.

**Definition 5.26** (Rationality Function)**.** *For a weight update function $f$, let $\rho_f : [0,1] \to [0,1]$ be the function from beliefs to predictions, such that reporting $\rho_f(b)$ is rational for an expert with belief $b$.*

We restrict our attention here on rationality functions that are proper functions, meaning that each belief leads to a single prediction. Note that for incentive compatible weight update functions, the rationality function is simply the identity function.

Under mild technical conditions on the rationality function, we show our main lower bound for (potentially non-IC) algorithms.[11]

**Theorem 5.27.** *For a weight update function $f$ with continuous or non-strictly increasing rationality function $\rho_f$, there is no deterministic no 2-regret algorithm.*

Note that Theorem 5.27 covers the standard algorithm, as well as other common update rules such as the Hedge update rule $f_{\text{Hedge}}(p_i^{(t)}, r^{(t)}) = e^{-\eta|p_i^{(t)} - r^{(t)}|}$ [Freund and Schapire, 1997], and all IC methods, since they have the identity rationality function (though the bounds in Thm 5.17 are stronger).

We start with a proof that any algorithm with non-strictly increasing rationality function must have worst-case loss strictly more than twice the best expert in hindsight. Conceptually, the proof is a generalization of the proof for Proposition 5.13.

**Lemma 5.28.** *Let $f$ be a weight update function with a non-strictly increasing rationality function $\rho_f$, such that there exists $b_1 < b_2$ with $\rho_f(b_1) \geq \rho_f(b_2)$. For every deterministic algorithm, in the worst case*

$$M^{(T)} \geq (2 + |b_2 - b_1|) m_i^{(T)}.$$

*Proof.* Fix, $f$, $b_1$ and $b_2$ such that $\rho_f(b_1) \geq \rho_f(b_2)$ with $b_1 < b_2$. Let $\pi_1 = \rho_f(b_1)$, $\pi_2 = \rho_f(b_2)$, $b_0 = 1 - \frac{b_2 + b_1}{2}$, and $\pi_0 = \rho_f(b_0)$.

[10] Similarly to Price of Anarchy (PoA) bounds, e.g. [Roughgarden and Tardos, 2007], the results here show the harm of selfish behavior. Unlike PoA bounds, we sidestep the question of equilibrium concepts and our results are additive rather than multiplicative.

[11] This holds even when the learning rate is parameterized similarly to Definition 5.15, as the rationality function does not change for different learning rates due to the linearity of the expectation operator.

Let there be two experts $e_0$ and $e_1$. Expert $e_0$ always predicts $\pi_0$ with belief $b_0$. If $\pi_1 = \pi_2$, $e_1$ predicts $\pi_1$ (similar to Proposition 5.13, we first fix the predictions of $e_1$, and will give consistent beliefs later). Otherwise $\pi_1 > \pi_2$, and expert $e_1$ has the following beliefs (and corresponding predictions) at time $t$:

$$b_1^{(t)} = \begin{cases} b_1 & \text{if } \frac{w_0^{(t)}\pi_0 + w_1^{(t)}\pi_2}{w_0^{(t)} + w_1^{(t)}} \geq \frac{1}{2} \\ b_2 & \text{otherwise} \end{cases}$$

The realizations are opposite of the algorithm's decisions.

We now fix the beliefs of $e_1$ in the case that $\pi_1 = \pi_2$. Whenever $r^{(t)} = 1$, set expert $e_1$'s belief to $b_2$, and whenever $r^{(t)} = 0$, set her belief to $b_1$. Note that the beliefs indeed lead to the predictions she made, by the fact that $\pi_1 = \rho_f(b_1) = \rho_f(b_2)$.

For the case where $\pi_1 > \pi_2$, if $(w_0^{(t)}\pi_0 + w_1^{(t)}\pi_2)/(w_0^{(t)} + w_1^{(t)}) \geq \frac{1}{2}$ then $e_1$'s belief will be $b_1$ leading to a report of $\pi_1$ and as $\pi_1 > \pi_2$ it must hold that $(w_0^{(t)}\pi_0 + w_1^{(t)}\pi_1)(w_0^{(t)} + w_1^{(t)}) > \frac{1}{2}$, hence the algorithm will certainly choose 1, so the realization is 0. Conversely, if $(w_0^{(t)}\pi_0 + w_1^{(t)}\pi_2)(w_0^{(t)} + w_1^{(t)}) < \frac{1}{2}$, then the belief of $e_1$ will be $b_2$ and her report will lead the algorithm to certainly choose 0, so the realization is 1. So in all cases, if the realization is 1, then the belief of expert $e_1$ is $b_2$ and otherwise it is $b_1$.

The total number of mistakes $M^{(T)}$ for the algorithm after $T$ time steps is $T$ by definition. Every time the realization was 1, $e_0$ will incur loss of $\frac{b_1+b_2}{2}$ and $e_1$ incurs a loss of $1 - b_2$, for a total loss of $1 - b_2 + \frac{b_1+b_2}{2} = 1 - \frac{b_2-b_1}{2}$. Whenever the realization was 0, $e_0$ incurs a loss of $1 - \frac{b_1+b_2}{2}$ and $e_1$ incurs a loss of $b_1$ for a total loss of $1 - \frac{b_1+b_2}{2} + b_1 = 1 - \frac{b_2-b_1}{2}$.

So the total loss for *both* of the experts is $\left(1 - \frac{b_2-b_1}{2}\right) \cdot T$, so the best expert in hindsight has $m_i^{(T)} \leq \frac{1}{2}\left(1 - \frac{b_2-b_1}{2}\right) \cdot T$. Rewriting yields the claim. □

For continuous rationality functions, we can generalize the results in Section 5.4 using a type of simulation argument. First, we address some edge cases.

**Proposition 5.29.** *For a weight update function $f$ with continuous strictly increasing rationality function $\rho_f$,*

1. *the regret is unbounded unless $\rho_f(0) < \frac{1}{2} < \rho(1)$; and*
2. *if $\rho_f(b) = \frac{1}{2}$ for $b \neq \frac{1}{2}$, the worst-case loss of the algorithm satisfies $M^{(T)} \geq (2 + |b - 1/2|)\, m_i^{(T)}$.*

*Proof.* First, assume that it does not hold that $\rho_f(0) < \frac{1}{2} < \rho_f(1)$. Since $\rho_f(0) < \rho_f(1)$ by virtue of $\rho_f$ being strictly increasing, it must be that either $\frac{1}{2} \leq \rho_f(0) < \rho_f(1)$

or $\rho_f(0) < \rho_f(1) \leq \frac{1}{2}$. Take two experts with $b_1^{(t)} = 0$ and $b_2^{(t)} = 1$. Realizations are opposite of the algorithm's predictions. Even though the experts have opposite beliefs, their predictions agree (potentially with one being indifferent), so the algorithm will consistently pick the same prediction, whereas one of the two experts will never make a mistake. Therefore the regret is unbounded.

As for the second statement. Since $\rho_f(0) < \frac{1}{2} < \rho_f(1)$, there is some $b$ such that $\rho_f(b) = \frac{1}{2}$. Wlog, assume $b < \frac{1}{2}$ (the other case is analogous). Since $\rho_f$ is continuous and strictly increasing, $\rho_f(\frac{b+1/2}{2}) > \frac{1}{2}$ while $\frac{b+1/2}{2} < \frac{1}{2}$. Take one expert $e_1$ with belief $b^{(t)} = \frac{b+1/2}{2} < \frac{1}{2}$, who will predict $p^{(t)} = \rho_f(\frac{b+1/2}{2}) > \frac{1}{2}$. Realizations are opposite of the algorithms decisions, and the algorithms decision is consistently 1, due to there only being one expert, and that expert putting more weight on 1. However, the absolute loss of the expert is only $\frac{1}{2} - \frac{|b-1/2}{2}$ at each time step. Summing over the timesteps and rewriting yields the claim. □

We are now ready to prove the main result in this section. The proof gives lower bound constants that are similar (though not identical) to the constants given in Lemmas 5.20, 5.23 and 5.25, though due to a reparameterization the factors are not immediately comparable. The proof appears in the appendix.

**Theorem 5.30.** *For a weight update function $f$ with continuous strictly increasing rationality function $\rho_f$, with $\rho_f(0) < \frac{1}{2} < \rho_f(1)$ and $\rho_f(\frac{1}{2}) = \frac{1}{2}$, there is no deterministic no 2-regret algorithm.*

Theorem 5.27 now follows from Lemma 5.28, Proposition 5.29 and Theorem 5.30.

## 5.6 Randomized Algorithms: Upper and Lower Bounds

### 5.6.1 Impossibility of Vanishing Regret

We now consider randomized online learning algorithms, which can typically achieve better worst-case guarantees than deterministic algorithms. For example, with honest experts, there are randomized algorithms that have a worst-case loss of $M^{(T)} \leq \left(1 + O\left(\frac{1}{\sqrt{T}}\right)\right) m_i^{(T)}$, which means that the regret with respect to the best expert in hindsight is vanishing as $T \to \infty$. Unfortunately, the lower bounds in Sections 5.4 and 5.5 below imply that no such result is possible for randomized algorithms.

**Corollary 5.31.** *Any incentive compatible randomized weight-update algorithm or non-IC randomized algorithm with continuous or non-strictly increasing rationality function cannot be no 1-regret.*

*Proof.* We can use the same instances as for Theorems 5.17 and 5.30 and Lemma 5.28 (whenever the algorithm was indifferent, the realizations were defined using the algorithm's tie-breaker rule; in the current setting simply pick any realization, say $r^t = 1$).

Whenever the algorithm made a mistake, it was because $\sum_i w_i^t s_i^t \geq \frac{1}{2}\sum_i w_i^t$. Therefore, in the randomized setting, it will still incur an expected loss of at least $\frac{1}{2}$. Therefore the total expected loss of the randomized algorithm is at least half that of the deterministic algorithm. Since the approximation factor for the latter is bounded away from 2 in all cases in Theorems 5.17 and 5.30 and Lemma 5.28, in these cases the worst-case loss of a randomized algorithm satisfies $M^{(T)} \geq (1 + \Omega(1))m_i^{(T)}$. □

### 5.6.2 An IC Randomized Algorithm for Selfish Experts

While we cannot hope to achieve a no-regret algorithm for online prediction with selfish experts, we can do better than the deterministic algorithm from Section 5.3. We now focus on the more general class of algorithms where the algorithm can make any prediction $q^{(t)} \in [0, 1]$ and incurs a loss of $|q^{(t)} - r^{(t)}|$. We give a randomized algorithm based on the Brier strictly proper scoring rule with loss at most 2.62 times that of the best expert as $T \to \infty$.

Perhaps the most natural choice for a randomized algorithm is to simply report a prediction of $q^{(t)} = \sum_{i=1}^n w_i^{(t)} p_i^{(t)} / \sum_{j=1}^n w_j^{(t)}$. However, this is problematic when the experts are highly confident and correct in their predictions. By the definition of a (bounded) strictly proper scoring rule, $\frac{d}{dp_i^{(t)}} f(p_i^{(t)}, 1)$ is 0 at 1 (and similarly the derivative is 0 around 0 for a realization of 0). This means that experts that are almost certain and correct will not have their weight reduced meaningfully, and so the proof that uses the potential function does not go through.

This motivates looking for an algorithm where the sum of weights of experts is guaranteed to decrease significantly whenever the algorithm incurs a loss. Consider the following generalization of RWM that rounds predictions to the nearest integer if they are with $\theta$ of that integer.

**Definition 5.32** ($\theta$-randomized weighted majority)**.** *Let $\mathcal{A}_r$ be the class of algorithms that maintains expert weights as in Definition 5.2. Let $b^{(t)} = \sum_{i=1}^n \frac{w_i^{(t)}}{\sum_{j=1}^n w_j^{(t)}} \cdot p_i^{(t)}$ be the weighted predictions. For parameter $\theta \in [0, \frac{1}{2}]$ the algorithm chooses 1 with probability*

$$p^{(t)} = \begin{cases} 0 & \text{if } b^{(t)} \leq \theta \\ b^{(t)} & \text{if } \theta < b^{(t)} \leq 1 - \theta \\ 1 & \text{otherwise.} \end{cases}$$

We call algorithms in $\mathcal{A}_r$ $\theta$-RWM algorithms. We'll use a $\theta$-RWM algorithm with the Brier rule. Recall that $s_i^{(t)} = |p_i^{(t)} - r^{(t)}|$; the Brier rule is defined as:

$$f_{\mathrm{Br}}(p_i^{(t)}, r^{(t)}) = 1 - \eta \left( \frac{(p_i^{(t)})^2 + (1 - p_i^{(t)})^2 + 1}{2} - (1 - s_i^{(t)}) \right). \tag{5.4}$$

**Theorem 5.33.** *Let $A \in \mathcal{A}_r$ be a $\theta$-RWM algorithm with the Brier weight update rule $f_{Br}$ and $\theta = 0.382$ and with $\eta = O(1/\sqrt{T}) \in (0, \frac{1}{2})$. $A$ has no 2.62-regret.*

The proof appears in the appendix.

## 5.7 Simulations

The theoretical results presented so far indicate that when faced with selfish experts, one should use an IC weight update rule, and ones with smaller scoring rule gap are better. Two objections to these conclusions are: first, the presented results are *worst-case*, and different instances are used to obtain the bounds for different scoring rules. A priori it is not obvious that for an arbitrary (non worst-case) input, the regret of different scoring rules follow the same relative ordering. It is of particular interest to see if the non-IC standard weight-update rule does better or worse than the IC methods proposed in this paper. Second, there is a gap between our upper and lower bounds for IC rules. It is therefore informative to look at different instances for which we expect our algorithms to do badly, to see if the performance is closer to the upper bound or to the lower bound.

### 5.7.1 Data-Generating Processes

To address these two concerns, we look at three different data-generating processes.

**Hidden Markov Model.** The experts are represented by a simple two-state hidden Markov model (HMM) with a "good" state and a "bad" state. We first flip a fair coin to determine the realization $r^{(t)}$. For $r^{(t)} = 0$ (otherwise beliefs are reversed), in the good state expert $i$ believes $b_i^{(t)} \sim \min\{\mathrm{Exp}(1)/5, 1\}$: the belief is exponentially distributed with parameter $\lambda = 1$, values are rescaled by $\frac{1}{5}$ and clamped between 0 and 1. In the bad state, expert $i$ believes $b_i^{(t)} \sim \mathrm{U}[\frac{1}{2}, 1]$. The transition probabilities to move to the other state are $\frac{1}{10}$ for both states. This data generating process models that experts that have information about the event are more accurate than experts who lack the information.

**Lower Bound Instance.** The lower bound instance described in the proof of Lemma 5.20.

**Greedy Lower Bound.** A greedy version of the lower bound described the proof of Lemma 5.20. There are 3 experts, one ($e_0$) who is mostly uninformative, and two ($e_1$ and $e_2$) who are alternating correct and incorrect. Whenever the weight of $e_0$ is "sufficiently" higher than that of $e_1$ and $e_2$, we have "punish the algorithm" by making $e_0$ wrong twice: $b_0^{(t)} = 0$, $b_1^{(t)} = 1$, $b_2^{(t)} = \frac{1}{2}$, $r^{(t)} = 1$, and $b_0^{(t+1)} = 0$, $b_1^{(t)} = \frac{1}{2}$, $b_0^{(t)} = 1$, $r^{(t)} = 1$. "Sufficiently" here means that weight of $e_0$ is high enough for the algorithm to follow its advice during both steps.

### 5.7.2 Results

**Hidden Markov Model Data.** In Figure 5.1 we show the regret as a function of time for the standard weight-update function, the Brier scoring rule, the spherical scoring rule, and a scoring rule from the Beta family [Buja et al., 2005] with $\alpha = \beta = \frac{1}{2}$. The expert's report $p_i^{(t)}$ for the IC methods correspond to their belief $b_i^{(t)}$, whereas for the standard weight-update rule, the expert reports $p_i^{(t)} = 1$ if $b_i^{(t)} \geq \frac{1}{2}$ and $p_i^{(t)} = 0$ otherwise. The $y$ axis is the ratio of the total loss of each of the algorithms to the performance of the best expert at that time. For clarity, we include the performance of the best expert at each time step, which by definition is 1 everywhere. The plot is for 10 experts, $T = 10,000$, $\eta = 10^{-2}$, and the randomized[12] versions of the algorithms (we return to why in a moment), averaged over 30 runs.

From the plot, we can see that each of the IC methods does significantly better than the standard weight-update algorithm. Whereas the standard weight-update rule levels off between 1.15 and 1.2, all of the IC methods dip below a regret of 1.05 at $T = 2,000$ and hover around 1.02 at $T = 10,000$. This trend continues and at $T = 200,000$ (not shown in the graph), the IC methods have a regret of about 1.003, whereas the regret for the standard algorithm is still 1.14. This gives credence to the notion that failing to account for incentive issues is problematic beyond the worst-case bounds presented earlier.

Moreover, the plot shows that while there is a worst-case lower bound for the IC methods that rules out no-regret, for quite natural synthetic data, the loss of all the IC algorithms approaches that of the best expert in hindsight, while the standard algorithm fails to do this. It curious to note that the performance of all IC methods are comparable (at least for this data-generating process). This seems to indicate that eliciting the truthful beliefs of the experts is more important than the exact weight-update rule.

Finally, note that the results shown here are for randomized weighted majority, using the different weight-update rules. For the deterministic version of the algorithms the difference between the non-IC standard weight-update rules and the IC ones

[12]Here we use the regular RWM algorithm, so in the notation of Section 5.6 we have $\theta = 0$.

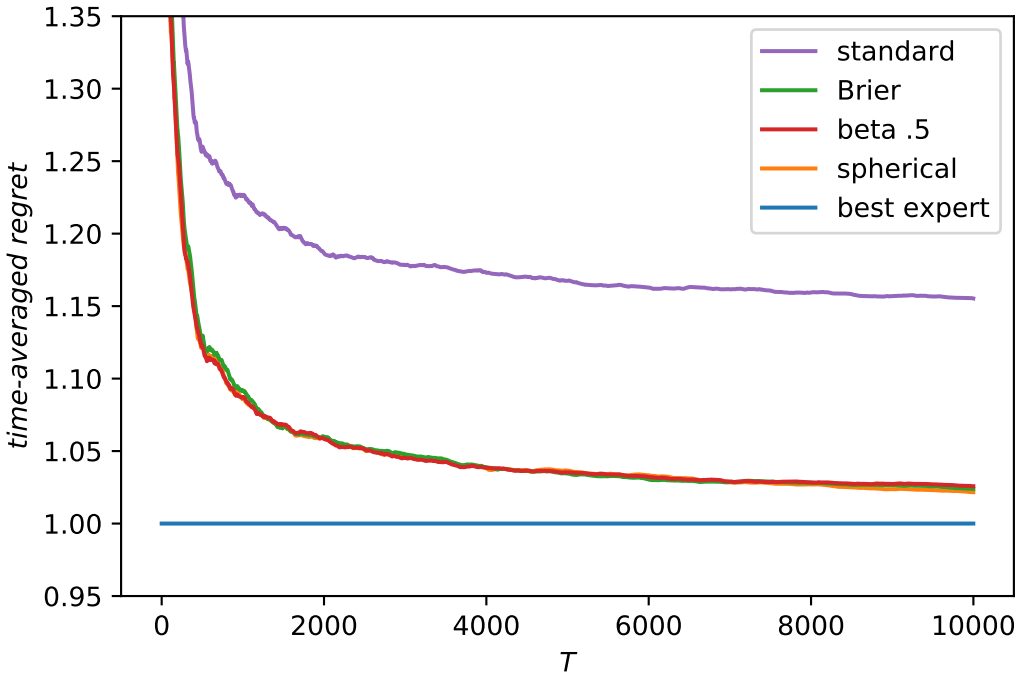



Figure 5.1: The time-averaged regret for the HMM data-generating process.

is even starker. Different choices for the transition probabilities of the HMM, and different distributions, e.g. the bad state has $b_i^{(t)} \sim \mathrm{U}[0,1]$, give similar results to the ones presented here.

**Comparison of LB Instances.** Now let's focus on the performance of different IC algorithms. First, in Figure 5.2 we show the regret for different algorithms on the greedy lower bound instance. Note that this instance is different from the one used in the proof of Lemma 5.20, but the regret is very close to what is obtained there. In fact, when we look at Table 5.1, we can see that very closely traces $2+\gamma$. In Table 5.1 we

Table 5.1: Comparison of lower bound results with simulation. The simulation is run for $T = 10,000, \eta = 10^{-4}$ and we report the average of 30 runs. For the lower bounds, the first number is the lower bound from Lemma 5.20, i.e. $2 + \frac{1}{\lceil \gamma^{-1} \rceil}$, the second number (in parentheses) is $2+\gamma$.

| | Beta .1 | Beta .5 | Beta .7 | Beta .9 | Brier | Spherical |
|---|---|---|---|---|---|---|
| Greedy LB | 2.3708 | 2.2983 | 2.2758 | 2.2584 | 2.2507 | 2.2071 |
| LB Sim | 2.4414 | 2.3186 | 2.2847 | 2.2599 | 2.2502 | 2.2070 |
| Lem 5.20 LB | 2.33 (2.441) | 2.25 (2.318) | 2.25 (2.285) | 2.25 (2.260) | 2.25 | 2.2 (2.207) |

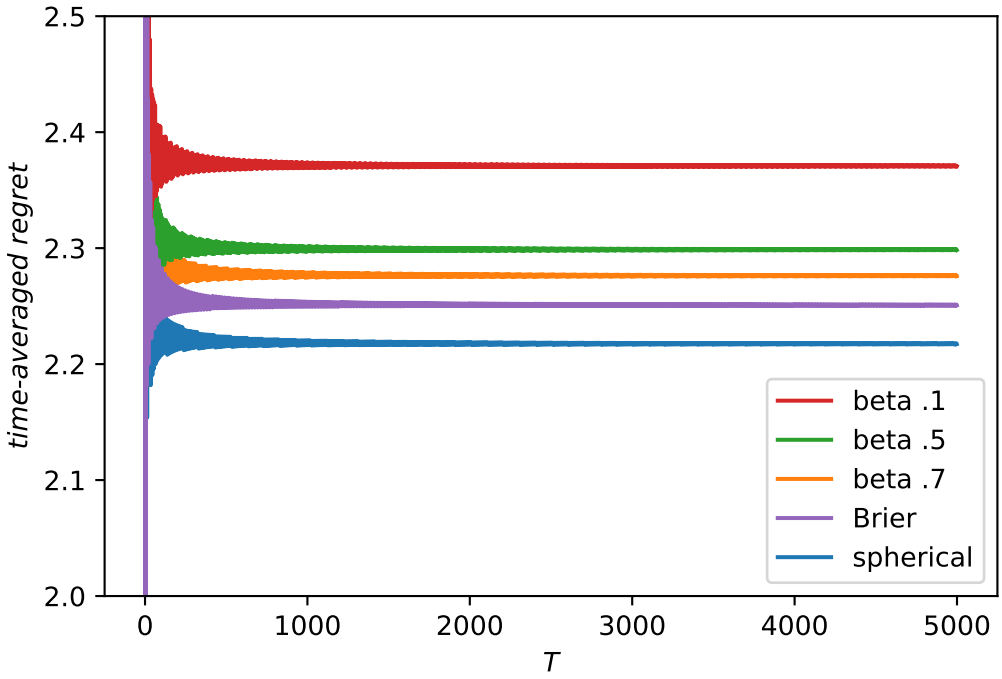



Figure 5.2: Regret for the greedy lower bound instance.

can also see the numerical results for the lower bound from Lemma 5.20. In fact, for the analysis, we needed to use $\lceil \gamma^{-1} \rceil$ when determining the first phase of the instance. When we use $\gamma$ instead numerically, the regret seems to trace $2+\gamma$ quite closely, rather than the weaker proven lower bound of $2 + \frac{1}{\lceil \gamma^{-1} \rceil}$. By using two different lower bound constructions, we can see that the analysis of Lemma 5.20 is essentially tight (up to the rounding of $\gamma$), though this does not exclude the possibility that stronger lower bounds are possible using more properties of the scoring rules (rather than only the scoring rule gap $\gamma$). In these experiments (and others we have performed), the regret of IC methods never exceeds the lower bound we proved in Lemma 5.20. Closing the gap between the lower and upper bound requires finding a different lower bound instance, or a better analysis for the upper bound.

# Chapter Appendix

## 5.A Omitted Proofs

### 5.A.1 Proof of Theorem 5.12

The WM algorithm with weight-update rule (5.2) for $\eta = O(1/\sqrt{T}) < \frac{1}{2}$ has no $(2+\sqrt{2})$-regret.

*Proof.* We use an intermediate potential function $\Phi^{(t)} = \sum_i w_i^{(t)}$. Whenever the algorithm incurs a loss, the potential must decrease substantially. For the algorithm incur a loss, it must have picked the wrong outcome. Therefore it loss $|r^{(t)} - t^{(t)}| = 1$ and $\sum_i w_i^{(t)} s_i^{(t)} \geq \frac{1}{2} \cdot \Phi^{(t)}$. We use this to show that in those cases the potential drops significantly:

$$
\begin{aligned}
\Phi^{(t+1)} &= \sum_i \left(1 - \eta\left(1 - \frac{1 - s_i^{(t)}}{\sqrt{\left(p_i^{(t)}\right)^2 + \left(1 - p_i^{(t)}\right)^2}}\right)\right) \cdot w_i^{(t)} \\
&\leq \sum_i \left(1 - \eta\left(1 - \sqrt{2}\left(1 - s_i^{(t)}\right)\right)\right) \cdot w_i^{(t)} && \left(\text{since } \min_x x^2 + (1-x)^2 = \frac{1}{2}\right) \\
&= (1-\eta)\,\Phi^{(t)} + \sqrt{2}\eta \sum_i \left(1 - s_i^{(t)}\right) w_i^{(t)} \\
&\leq (1-\eta)\,\Phi^{(t)} + \frac{\sqrt{2}\eta}{2}\Phi^{(t)} && \left(\text{since } \sum_i w_i^{(t)}\left(1 - s_i^{(t)}\right) \leq \frac{1}{2}\Phi^{(t)}\right) \\
&= \left(1 - \frac{2-\sqrt{2}}{2}\eta\right)\Phi^{(t)}
\end{aligned}
$$

Since initially $\Phi^0 = n$, after $M^{(T)}$ mistakes, we have:

$$\Phi^T \leq n\left(1 - \frac{2-\sqrt{2}}{2}\eta\right)^{M^{(T)}}. \tag{5.5}$$

Now, let's bound the final weight of expert $i$ in terms of the number of mistakes she made:

$$\begin{aligned}
w_i^{(T)} &= \prod_t \left(1 - \eta\left(1 - \frac{1 - s_i^{(t)}}{\sqrt{\left(p_i^{(t)}\right)^2 + \left(1 - p_i^{(t)}\right)^2}}\right)\right) \\
&\geq \prod_t \left(1 - \eta s_i^{(t)}\right) && \left(\text{since } \max_{x\in[0,1]} x^2 + (1-x)^2 = 1\right) \\
&\geq \prod_t (1-\eta)^{s_i^{(t)}} && (\text{since } 1 - \eta x \geq (1-\eta)^x \text{ for } x \in [0,1]) \\
&= (1-\eta)^{\sum_t s_i^{(t)}} \\
&= (1-\eta)^{m_i^{(T)}}
\end{aligned}$$

Combining this with $w_i^{(t)} \leq \Phi^{(t)}$ and (5.5), and taking natural logarithms of both sides we get:

$$\begin{aligned}
\ln\left((1-\eta)^{m_i^{(T)}}\right) &\leq \ln\left(n\left(1 - \frac{2-\sqrt{2}}{2}\eta\right)^{M^{(T)}}\right) \\
m_i^{(T)} \cdot \ln(1-\eta) &\leq M^{(T)} \cdot \ln\left(1 - \frac{2-\sqrt{2}}{2}\eta\right) + \ln n \\
m_i^{(T)} \cdot \left(-\eta - \eta^2\right) &\leq M^{(T)} \cdot \ln\left(\exp\left(-\frac{2-\sqrt{2}}{2}\eta\right)\right) + \ln n \\
m_i^{(T)} \cdot \left(-\eta - \eta^2\right) &\leq M^{(T)} \cdot -\frac{2-\sqrt{2}}{2}\eta + \ln n \\
M^{(T)} &\leq \left(\frac{2}{2-\sqrt{2}}\right) \cdot \left((1+\eta)m_i^{(T)} + \frac{\ln n}{\eta}\right)
\end{aligned}$$

where in the third inequality we used $-\eta - \eta^2 \leq \ln(1-\eta)$ for $\eta \in (0, \frac{1}{2})$. Rewriting the last statement proves the claim. □

### 5.A.2 Proof of Lemma 5.20

Let $\mathcal{F}$ be a family of scoring rules generated by a symmetric strictly proper scoring rule $f$, and let $\gamma$ be the scoring rule gap of $\mathcal{F}$. In the worst case, MW with any scoring rule $f' \in \mathcal{F}$ with $\eta \in (0, \frac{1}{2})$, algorithm loss $M^{(T)}$ and expert loss $m_i^{(T)}$, satisfies

$$M^{(T)} \geq \left(2 + \frac{1}{\lceil \gamma^{-1} \rceil}\right) \cdot m_i^{(T)}.$$

*Proof.* Let $a$, $\eta$ be the parameters of $f'$ in the family $\mathcal{F}$, as in Definition 5.15. Fix $T$ sufficiently large and an integer multiple of $2\lceil \gamma^{-1} \rceil + 1$, and let $e_1$, $e_2$, and $e_3$ be three experts. For $t = 1, ..., \alpha \cdot T$ where $\alpha = \frac{2\lceil \gamma^{-1} \rceil}{2\lceil \gamma^{-1} \rceil + 1}$ such that $\alpha T$ is an even integer, let $p_1^{(t)} = \frac{1}{2}$, $p_2^{(t)} = 0$, and $p_3^{(t)} = 1$. Fix any tie-breaking rule for the algorithm. Realization $r^{(t)}$ is always the opposite of what the algorithm chooses.

Let $M^{(t)}$ be the loss of the algorithm up to time $t$, and let $m_i^{(t)}$ be the loss of expert $i$. We first show that at $t' = \alpha T$, $m_1^{(t')} = m_2^{(t')} = m_3^{(t')} = \frac{\alpha T}{2}$ and $M^{(t')} = \alpha T$. The latter part is obvious as $r^{(t)}$ is the opposite of what the algorithm chooses. That $m_1^{(t')} = \frac{\alpha T}{2}$ is also obvious as it adds a loss of $\frac{1}{2}$ at each time step. To show that $m_2^{(t')} = m_3^{(t')} = \frac{\alpha T}{2}$ we do induction on the number of time steps, in steps of two. The induction hypothesis is that after an even number of time steps, $m_2^{(t)} = m_3^{(t)}$ and that $w_2^{(t)} = w_3^{(t)}$. Initially, all weights are 1 and both experts have loss of 0, so the base case holds. Consider the algorithm after an even number $t$ time steps. Since $w_2^{(t)} = w_3^{(t)}$, $p_3^{(t)} = 1 - p_2^{(t)}$, and $p_1^{(t)} = \frac{1}{2}$ we have that $\sum_{i=1}^3 w_i^{(t)} p_i^{(t)} = \sum_{i=1}^3 w_i^{(t)}(1 - p_i^{(t)})$ and hence the algorithm will use its tie-breaking rule for its next decision. Thus, either $e_2$ or $e_3$ is wrong. Wlog let's say that $e_2$ was wrong (the other case is symmetric), so $m_2^{(t+1)} = 1 + m_3^{(t+1)}$. Now at time $t+1$, $w_2^{(t+1)} = (1-\eta) w_3^{(t+1)} < w_3^{(t+1)}$. Since $e_1$ does not express a preference, and $e_3$ has a higher weight than $e_2$, the algorithm will follow $e_3$'s advice. Since the realization $r^{(t+1)}$ is the opposite of the algorithms choice, this means that now $e_3$ incurs a loss of one. Thus $m_2^{(t+2)} = m_2^{(t+1)}$ and $w_2^{(t+2)} = w_2^{(t+1)}$ and $m_3^{(t+2)} = 1 + m_3^{(t+1)} = m_2^{(t+2)}$. The weight of expert $e_2$ is $w_2^{(t+2)} = aa(1-\eta) w_2^{(t)}$ and the weight of expert $e_3$ is $w_3^{(t+2)} = a(1-\eta) a w_3^{(t)}$. By the induction hypothesis $w_2^{(t)} = w_3^{(t)}$, hence $w_2^{(t+2)} = w_3^{(t+2)}$, and since we already showed that $m_2^{(t+2)} = m_3^{(t+2)}$, this completes the induction.

Now, for $t = \alpha T + 1, ..., T$, we let $p_1^{(t)} = 1$, $p_2^{(t)} = 0$, $p_3^{(t)} = \frac{1}{2}$ and $r^{(t)} = 0$. So henceforth $e_3$ does not provide information, $e_1$ is always wrong, and $e_2$ is always right. If we can show that the algorithm will always follow $e_1$, then the best expert is $e_2$ with a loss of $m_2^{(T)} = \frac{\alpha T}{2}$, while the algorithm has a loss of $M^{(T)} = T$. If this holds for $\alpha < 1$ this proves the claim. So what's left to prove is that the algorithm will always follow $e_1$. Note that since $p_3^{(t)} = \frac{1}{2}$ it contributes equal amounts

to $\sum_{i=1}^{3} w_i^{(t)} p_i^{(t)}$ and $\sum_{i=1}^{3} w_i^{(t)}(1-p_i^{(t)})$ and is therefore ignored by the algorithm in making its decision. So it suffices to look at $e_1$ and $e_2$. The algorithm will pick 1 iff $\sum_{i=1}^{3} w_i^{(t)}(1-p_i^{(t)}) \leq \sum_{i=1}^{3} w_i^{(t)} p_i^{(t)}$, which after simplifying becomes $w_1^{(t)} > w_2^{(t)}$.

At time step $t$, $w_1^{(t)} = \big(a(1+\eta(f(\frac{1}{2})-1))\big)^{\alpha T} (a\cdot(1-\eta))^{t-\alpha T}$ for expert $e_1$ and $w_2^{(t)} = (a(1-\eta))^{\frac{\alpha T}{2}}\, a^{\frac{\alpha T}{2}+t-\alpha T}$ for expert $e_2$. We have that $w_1^{(t)}$ is decreasing faster in $t$ than $w_2^{(t)}$. So if we can show that $w_1^{(T)} \geq w_2^{(T)}$ for some $\alpha < 1$, then $e_2$ will incur a total loss of $\alpha T/2$ while the algorithm incurs a loss of $T$ and the statement is proved.

We have that $w_1^{(t)}$ is decreasing faster in $t$ than $w_2^{(t)}$. So if we can show that at time $T$, $w_1^{(T)} \geq w_2^{(T)}$ for some $\alpha < 1$, then $e_2$ will incur a total loss of $\alpha T$ while the algorithm incurs a loss of $T$ and the statement is proved. First divide both weights by $a^T$ so that we have

$$\begin{aligned} a^{-T} w_1^{(T)} &= \big(1+\eta(f(\tfrac{1}{2})-1)\big)^{\alpha T} (1-\eta)^{(1-\alpha)T} \\ a^{-T} w_2^{(T)} &= (1-\eta)^{\frac{\alpha T}{2}}. \end{aligned}$$

Let $\alpha = \frac{2\lceil\gamma^{-1}\rceil}{2\lceil\gamma^{-1}\rceil+1}$ and recall that $T = k\cdot(2\lceil\gamma^{-1}\rceil+1)$ for positive integer $k$. Thus we can write

$$\begin{aligned} a^{-T} w_1^{(T)} &= \big(1+\eta(f(\tfrac{1}{2})-1)\big)^{k2\lceil\gamma^{-1}\rceil} (1-\eta)^{k} \\ &= \Big(\big(1+\eta(f(\tfrac{1}{2})-1)\big)^{2\lceil\gamma^{-1}\rceil} (1-\eta)\Big)^{k} \\ a^{-T} w_2^{(T)} &= (1-\eta)^{k\lceil\gamma^{-1}\rceil} \\ &= \Big((1-\eta)^{\lceil\gamma^{-1}\rceil}\Big)^{k} \end{aligned}$$

So it holds that $w_1^{(T)} > w_2^{(T)}$ if we can show that $\big(1+\eta(f(\frac{1}{2})-1)\big)^{2\lceil\gamma^{-1}\rceil}(1-\eta) > (1-\eta)^{\lceil\gamma^{-1}\rceil}$

$$\begin{aligned}
\left(1+\eta(f(\tfrac{1}{2})-1)\right)^{2\lceil\gamma^{-1}\rceil}(1-\eta) &= \left(1-(\tfrac{1}{2}-\gamma)\eta\right)^{2\lceil\gamma^{-1}\rceil}(1-\eta) && \text{(def. of } \gamma) \\
&\geq (1-\eta+2\gamma\eta)^{\lceil\gamma^{-1}\rceil}(1-\eta) && (5.6) \\
&= \left(\frac{1-\eta+2\gamma\eta}{1-\eta}\right)^{\lceil\gamma^{-1}\rceil}(1-\eta)^{\lceil\gamma^{-1}\rceil+1} \\
&= (1+2\gamma\eta)^{\lceil\gamma^{-1}\rceil}(1-\eta)^{\lceil\gamma^{-1}\rceil+1} \\
&\geq \left(1+\lceil\gamma^{-1}\rceil 2\gamma\eta\right)(1-\eta)^{\lceil\gamma^{-1}\rceil+1} \\
&\geq \left((1+2\eta)(1-\eta)\right)(1-\eta)^{\lceil\gamma^{-1}\rceil} \\
&> (1-\eta)^{\lceil\gamma^{-1}\rceil} && (\text{for } \eta < \tfrac{1}{2})
\end{aligned}$$

Therefore expert $e_2$ will not incur any more loss during the last stage of the instance, so her total loss is $m_i^{(T)} = k\lceil\gamma^{-1}\rceil$ while the loss of the algorithm is $M^{(T)} = T = k\cdot(2\lceil\gamma^{-1}\rceil+1)$. So

$$\frac{M^{(T)}}{m_i^{(t)}} \geq \frac{k\cdot(2\lceil\gamma^{-1}\rceil+1)}{k\lceil\gamma^{-1}\rceil} = 2+\frac{1}{\lceil\gamma^{-1}\rceil}$$

rearranging proves the claim. □

### 5.A.3 Proof of Lemma 5.25

Let $\mathcal{F}$ be a family of scoring rules generated by a normalized strictly proper scoring rule $f$, with not both $f(0,0) = f(1,1)$ and $f(0,1) = f(1,0)$ and parameters $c$ and $d$ as in Definition 5.24. In the worst case, MW with any scoring rule $f'$ from $\mathcal{F}$ with $\eta \in (0, \frac{1}{2})$ can do no better than

$$M^{(T)} \geq \left(2+\max\{\tfrac{1-c}{2c}, \tfrac{d}{4(1-d)}\}\right)\cdot m_i^{(T)}.$$

*Proof.* Fix $f$, and without loss of generality assume that $f(0,0) = 1$ (since $f$ is normalized, either $f(0,0)$ or $f(1,1)$ needs to be 1, rename if necessary). As $f$ is normalized, at least one of $f(0,1)$ and $f(1,0)$ needs to be 0. For now, we consider the case where $f(0,1) = 0$, we treat the other case later. For now we have $f(0,0) = 1$, $f(0,1) = 0$, and by definition 5.24, $f(1,0) = 1-c$ and $f(1,1) = 1-d$, where $c > d$ (since correctly reporting 1 needs to score higher than reporting 0 when 1 materialized) and $\neg(c = 1 \wedge d = 0)$ (since that would put us in the semi-symmetric case).

We construct an instance as follows. We have two experts, $e_0$ reports 0 always,

and $e_1$ reports 1 always, and as usual, the realizations are opposite of the algorithms decisions. Since the experts have completely opposite predictions, the algorithm will follow whichever expert has the highest weight. We will show that after a constant number of time steps $t$, the weight $w_0^{(t)}$ of $e_0$ will be larger than the weight $w_1^{(t)}$ of $e_1$ even though $e_0$ will have made one more mistake. Note that when this is true for some $t$ independent of $\eta$, this implies that the algorithm cannot do better than $2\frac{t}{t-1} > 2 + \frac{2}{t}$.

While it hasn't been the case that $w_0^{(t)} > w_1^{(t)}$ with $m_0^{(t)} = m_1^{(t)} + 1$, realizations alternate, and the weight of each expert is:

$$\begin{aligned} w_0^{(2t)} &= a^{2t}(1+\eta(f(0,0)-1))^t(1+\eta(f(0,1)-1))^t \\ &= a^{2t}(1+\eta(1-1))^t(1+\eta(1-c-1))^t \\ &= a^{2t}(1-c\eta)^t \qquad (5.7) \\ w_1^{(2t)} &= a^{2t}(1+\eta(f(1,1)-1))^t(1+\eta(f(1,0)-1))^t \\ &= a^{2t}(1+\eta((1-d)-1))^t(1+\eta(0-1))^t \\ &= a^{2t}(1-d\eta)^t(1-\eta)^t \qquad (5.8) \end{aligned}$$

What remains to be shown is that for some $t$ independent of $\eta$,

$$a^{2t+1}(1-c\eta)^{t+1} > a^{2t+1}(1-d\eta)^{t+1}(1-\eta)^t.$$

We know that it cannot be the case that simultaneously $c = 1$ and $d = 0$, so let's first consider the case where $c < 1$. In this case, it is sufficient to prove the above statement assuming $d = 0$, as this implies the inequality for all $d \in [0, c)$. The following derivation shows that $a^{2t+1}(1-c\eta)^{t+1} > a^{2t+1}(1-d\eta)^{t+1}(1-\eta)^t$ whenever $\frac{c}{(1-c)} < t$.

$$
\begin{aligned}
a^{2t+1}(1-c\eta)^{t+1} &> a^{2t+1}(1-d\eta)^{t+1}(1-\eta)^t \\
(1-c\eta)^{t+1} &> (1-\eta)^t && (d=0) \\
(1-c\eta) &> \left(\frac{1-\eta}{1-c\eta}\right)^t \\
\ln(1-c\eta) &> t\cdot\ln\left(\frac{1-\eta}{1-c\eta}\right) \\
1-\frac{1}{1-c\eta} &> t\cdot\left(\frac{1-\eta}{1-c\eta}-1\right) && (1-\tfrac{1}{x}\le\ln x\le x-1) \\
\frac{1-c\eta-1}{1-c\eta} &> t\cdot\left(\frac{1-\eta-1+c\eta}{1-c\eta}\right) \\
\frac{c\eta}{1-c\eta} &< t\cdot\frac{(1-c)\eta}{1-c\eta} \\
\frac{c\eta}{(1-c)\eta} &< t \\
\frac{c}{(1-c)} &< t
\end{aligned}
$$

So after $2t+1$ time steps for some $t \le \frac{c}{1-c}+1$, expert $e_0$ will have one more mistake than expert $e_1$, but still have a higher weight. This means that after at most another $2t+1$ time steps, she will have two more mistakes, yet still a higher weight. In general, the total loss of the algorithm is at least $2+\frac{1-c}{c}$ times that of the best expert. Now consider the case where $c=1$ and therefore $d>0$. We will show that after $2t+1$ time steps for some $t \le 2\frac{1-d}{d}+1$ expert $e_0$ will have one more mistake than expert $e_1$.

$$
\begin{aligned}
a^{2t}(1-c\eta)^{t+1} &> a^{2t}(1-d\eta)^t(1-\eta)^t(1-d\eta) \\
(1-\eta)^{t+1} &> (1-d\eta)^{t+1}(1-\eta)^t && (c=1) \\
\frac{1-\eta}{1-d\eta} &> (1-d\eta)^t \\
\ln\left(\frac{1-\eta}{1-d\eta}\right) &> t\ln(1-d\eta) \\
1-\frac{1-d\eta}{1-\eta} &> t(1-d\eta-1) && (1-\tfrac{1}{x} \leq \ln x \leq x-1) \\
\frac{1-\eta-1+d\eta}{1-\eta} &> -td\eta \\
\frac{(1-d)\eta}{1-\eta} &< td\eta \\
\frac{1-d}{d}\frac{1}{1-\eta} &< t \\
2\frac{1-d}{d} &< t && (\text{by } \eta < \tfrac{1}{2})
\end{aligned}
$$

So in any case, after $t \leq 2\max\{\frac{c}{1-c}, \frac{1-d}{d}\} + 1$ time steps so the loss compared to the best expert is at least

$$2 + \max\{\tfrac{1-c}{c}, \tfrac{d}{2(1-d)}\}.$$

What remains to be proven is the case where $f(0,1) > 0$. In this case, it will have to be that $f(1,0) = 0$, as $f$ is normalized. And similarly to before, by Definition 5.24, we have $f(0,1) = 1-c$ and $f(1,1) = 1-d$ for $c > d$ and $\neg(c = 1 \wedge d = 0)$. Now, whenever $w_o^{(t)} > w_1(t)$, $e_0$ will predict 1 and $e_1$ predicts 0, and otherwise $e_0$ predicts 0 and $e_1$ predicts 1. As usual, the realizations are opposite of the algorithm's decisions. For now assume tie of the algorithm is broken in favor of $e_1$, then the weights will be identical to (5.7), (5.8). If the tie is broken in favor of $e_0$ initially, it takes at most twice as long before $e_0$ makes two mistakes in a row. Therefore, the loss with respect to the best expert in hindsight of an algorithm with any asymmetric strictly proper scoring rule is

$$2 + \max\{\tfrac{1-c}{2c}, \tfrac{d}{4(1-d)}\}.$$

□

### 5.A.4 Proof of Theorem 5.30

For a weight update function $f$ with continuous strictly increasing rationality function $\rho_f$, with $\rho_f(0) < \frac{1}{2} < \rho_f(1)$ and $\rho_f(\frac{1}{2}) = \frac{1}{2}$, there is no deterministic no 2-regret algorithm.

*Proof.* Fix $f$ with $\rho_f(0) < \frac{1}{2} < \rho_f(1)$ and $\rho_f(\frac{1}{2}) = \frac{1}{2}$. Define $p = \max\{\rho_f(0), 1-\rho_f(1)\}$, so that $p$ and $1-p$ are both in the image of $\rho_f$ and the difference between $p$ and $1-p$ is as large as possible. Let $b_1 = \rho^{-1}(p)$ and $b_2 = \rho^{-1}(1-p)$ and observe that $b_1 < \frac{1}{2} < b_2$.

Next, we rewrite the weight-update function $f$ in a similar way as the normalization procedure similar to Definition 5.15: $f(p, r) = a(1 + \eta(f'(p, r) - 1))$. where $\max\{f'(p, 0), f'(1-p, 1)\} = 1$ and $\min\{f'(p, 1), f'(1-p, 0)\} = 0$. Again we do this to prove bounds that are not dependent on any learning rate parameter.

Note that the composition of $\rho_f$ and $f$, namely $f(\rho_f(p), r)$ is a strictly proper scoring rule, since it changes the prediction in the same way as the selfish expert would do. Since $f(\rho_f(p), r)$, it must also be that $f'(\rho_f(p), r)$ is a strictly proper scoring rule, since it is a positive affine transformation of $f \circ \rho_f$.[13]

We now continue similarly to the lower bounds in Section 5.4. We only treat the semi-symmetric and asymmetric cases as the former includes the special case of the symmetric weight-update function.

For the semi-symmetric case, by definition $f'(\rho_f(b_1), 0) = f'(\rho_f(b_2), 1) = 1$ and $f'(p, 0), f'(1-p, 1)\} = 1$ and $\min\{f'(p, 1), f'(1-p, 0) = 0$. Because $f' \circ \rho_f$ is a strictly proper scoring rule, the following inequality holds:

$$\tfrac{1}{2}f'(\rho_f(\tfrac{1}{2}), 0) + \tfrac{1}{2}f'(\rho_f(\tfrac{1}{2}, 1)) + \mu = \tfrac{1}{2}f'(\rho_f(b_1), 0) + \tfrac{1}{2}f'(\rho_f(\tfrac{1}{2}, 1)) = \tfrac{1}{2}$$

for some $\mu > 0$, since an expert with belief $\rho_f(\frac{1}{2})$ must have a strict incentive to report this. Here $\mu$ plays the same role as the semi-symmetric scoring rule gap.[14]

We now pitch three experts against each other in a similar lower bound instance as Lemma 5.23. For the first stage, they have beliefs $b_0^{(t)} = \frac{1}{2}$, $b_1^{(t)} = b_1$, $b_2^{(t)} = b_2$, so they have predictions $p_0^{(t)} = \frac{1}{2}$, $p_1^{(t)} = \rho_f(b_1) = p$, $p_2^{(t)} = \rho_f(b_2) = 1 - p$. For the second stage, recall that either $b_1 = 0$ or $b_2 = 1$. In the former case, $b_0^{(t)} = 1$, $b_1^{(t)} = 0$, $b_2^{(t)} = \frac{1}{2}$ and $r^{(t)} = 0$ and in the latter case $b_0^{(t)} = 0$, $b_1^{(t)} = 1$, $b_2^{(t)} = \frac{1}{2}$ and $r^{(t)} = 1$. We now show a bijection between the instance in Lemma 5.23 and this instance, which establishes the lower bound for the semi-symmetric non-incentive compatible case. First of all, note that the weights of each of the experts in the first stage is the same

[13] And since $f'$ is a positive affine transformation of $f$, the rationality function is unchanged due to the linearity of the expectation operator.

[14] It is defined slightly differently though, as the image of $\rho_f$ may not be $[0, 1]$.

(up to the choice of $a$ and $\eta$, and for now assuming that the algorithms choices and thus the realizations are the same):

$$\begin{aligned} w_0^{(2t)} &= a^{2t}\left((1+\eta(f'(\tfrac{1}{2},0)-1))((1+\eta(f'(\tfrac{1}{2},1)-1))\right)^t \\ &\geq a^{2t}(1-\eta+2\mu\eta)^t && \text{(Follows from (5.3))} \\ w_1^{(2t)} &= a^{2t}\left(1-\eta)\right)^t \\ w_1^{(2t)} &= a^{2t}\left(1-\eta)\right)^t \end{aligned}$$

In the second stage expert $e_0$ is always wrong and $e_1$ is always right, and hence at time $T$ the weights

Also note, that the predictions of $e_1$ and $e_2$ are opposite, i.e. $p$ and $1-p$, so the algorithm will follow the expert which highest weight, meaning the algorithms decisions and the realizations are identical to the instance in Lemma 5.23.

To complete the proof of the lower bound instance, we need to show that the total loss of $e_1$ is the same. During the first stage, alternatingly the true absolute loss of $e_1$ is $b_1$ and $1-b_1$, so after each 2 steps, her loss is 1. During the last stage, since her belief is certain (i.e. $b_0$ if $b_1 = 0$ or $b_2$ if $b_2 = 1$) ans she is correct, she incurs no additional loss. Therefore the loss of the algorithm and the true loss of $e_1$ are the same as in Lemma 5.23, hence the loss of the algorithm is at least $\frac{1}{\lceil \mu^{-1} \rceil}$ times that of the best expert in hindsight.

Finally, we consider the asymmetric case. We use a similar instance as Lemma 5.25 with two experts $e_0$, $e_1$. If $f'(1-p,0) = 0$ we have $b_0^{(t)} = b_1$ and $b_1^{(t)} = b_2$, so $p_0^{(t)} = p$ and $p_1^{(t)} = 1-p$, otherwise the beliefs (and thus predictions) alternate. Again, the predictions are opposite of each other, and the weights evolve identically (up to the choice of $a$ and $\eta$) as before. Again the loss up until the moment that the same expert is chosen twice in a row is the same.

Once the same expert is chosen twice (after at most $2\max\{\frac{c}{1-c}, \frac{1-d}{d}\}+1$) steps), it is not necessarily the case that the total loss of one expert exceeds the other by 2, as the true beliefs are $b_1$ and $b_2$, rather than 0 and 1. However, since at least either $b_1 = 0$ or $b_2 = 1$, and $b_1 < \frac{1}{2} < b_2$, the difference in total absolute loss in this non-IC instance is at least half of the IC instance, so we lose at most factor $\frac{1}{2}$ in the regret bound, hence for the asymmetric case $M^{(T)} \geq \left(2 + \max\{\frac{1-c}{4c}, \frac{d}{8(1-d)}\}\right) m_i^{(t)}$, completing the proof of the statement. □

## 5.A.5 Proof of Theorem 5.33

Let $A \in \mathcal{A}_r$ be a $\theta$-RWM algorithm with the Brier weight update rule $f_{\mathrm{Br}}$ and $\theta = 0.382$ and with $\eta = O(1/\sqrt{T}) \in (0, \frac{1}{2})$. $A$ has no 2.62-regret.

*Proof.* The core difference between the proof of this statement, and the proof for Theorem 5.12 is in giving the upper bound of $\Phi^{(t+1)}$. Here we will give an upper bound of $\Phi^{(T)} \leq n \cdot \exp\left(-\frac{\eta}{2.62}M^{(T)}\right)$. Before giving this bound, observe that this would imply the theorem: since the weight updates are identical to the deterministic algorithm, we can use the same lower bound for $\Phi^{(T)}$, namely $\Phi^{(T)} \geq (1-\eta)^{m_i^{(T)}}$ for each expert. Then taking the log of both sides we get:

$$\begin{aligned}
\ln n - \frac{\eta}{2.62}M^{(T)} &\geq m_i^{(T)} \cdot \ln(1-\eta) \\
\ln n - \frac{\eta}{2.62}M^{(T)} &\geq m_i^{(T)} \cdot (-\eta - \eta^2) \\
M^{(T)} &\leq 2.62\left((1+\eta)m_i^{(T)} + \frac{\ln n}{\eta}\right)
\end{aligned}$$

So all that's left to prove is that whenever the algorithm incurs a loss $\ell$, $\Phi^{(t+1)} \leq \exp\left(-\frac{\eta}{2.62}\ell\right)$. At time $t$, the output $q^{(t)}$ of a $\theta$-RWM algorithm is one of three cases, depending on the weighted expert prediction. The first options is that the algorithm reported the realized event, in which case the $\ell^{(t)} = 0$ and the statement holds trivially. We treat the other two cases separately.

Let's first consider the case where the algorithm reported the incorrect event with certainty: $\ell^{(t)} = 1$. The means that $\sum_{i=1}^n w_i^{(t)} s_i^{(t)} \geq (1-\theta)\Phi^{(t)}$. Since the Brier rule is concave, $\Phi^{(t+1)}$ is maximized when $s_i^{(t)} = 1-\theta$ for all experts $i$. In this case each we get

$$\begin{aligned}
\Phi^{(t+1)} &\leq \sum_i \left(1 - \eta\left(\frac{(p_i^{(t)})^2 + (1-p_i^{(t)})^2 + 1}{2} - (1 - s_i^{(t)})\right)\right) w_i^{(t)} \\
&\leq \sum_i \left(1 - \eta\left(\frac{(\theta)^2 + (1-\theta)^2 + 1}{2} - \theta\right)\right) w_i^{(t)} \\
&\leq \sum_i \left(1 - \frac{\eta}{2.62}\right) w_i^{(t)} && \text{(since } \theta = .382\text{)} \\
&= \left(1 - \frac{\eta}{2.62}\ell^{(t)}\right)\Phi^{(t)}.
\end{aligned}$$

Otherwise the algorithms report is between $\theta$ and $1-\theta$. Let $\ell^{(t)} \in [\theta, 1-\theta]$ be the loss of the algorithm. Again, since the Brier rule is concave, $\Phi^{(t+1)}$ is maximized when $s_i^{(t)} = \ell^{(t)}$ for all experts $i$. On $[\theta, 1-\theta]$ the Brier proper scoring rule can be upper bounded by

$$1 - \frac{\eta}{f_{\mathrm{Br}}(1-\theta,1)/\theta}s_i^{(t)} = 1 - \frac{\eta}{2.62}s_i^{(t)}.$$

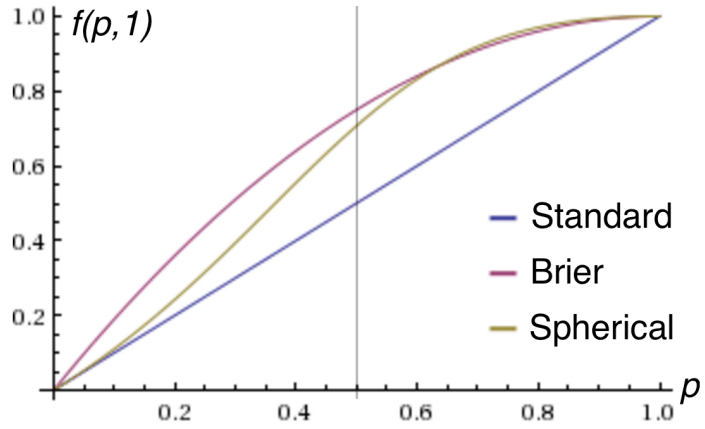



Figure 5.3: Three different normalized weight-update rules for $r^{(t)} = 1$. The line segment is the standard update rule, the concave curve the Brier rule and the other curve the spherical rule.

This yields

$$\begin{aligned}\Phi^{(t+1)} &\leq \sum_i \left(1 - \eta\left(\frac{(p_i^{(t)})^2 + (1-p_i^{(t)})^2 + 1}{2} - (1 - s_i^{(t)})\right)\right) w_i^{(t)} \\ &\leq \sum_i \left(1 - \frac{\eta}{2.62} s_i^{(t)}\right) w_i^{(t)} \\ &\leq \left(1 - \frac{\eta}{2.62}\ell^{(t)}\right) \Phi^{(t)}\end{aligned}$$

So the potential at time $T$ can be bounded by $\Phi^{(T)} \leq n \cdot \prod_t \left(1 - \frac{\eta}{2.62}\ell^{(t)}\right) \leq n \cdot \exp\left(-\frac{\eta}{2.62}M^{(T)}\right)$, from which the claim follows. □

## 5.B Selecting a Strictly Proper Scoring Rule

When selecting a strictly proper scoring rule for an IC online prediction algorithm, different choices may lead to very different guarantees. Many different scoring rules exist [McCarthy, 1956, Savage, 1971], and for discussion of selecting proper scoring rules in non-online settings, see also [Merkle and Steyvers, 2013]. Figure 5.3 shows two popular strictly proper scoring rules, the Brier rule and the spherical rule, along with the standard rule as comparison. Note that we have normalized all three rules for easy comparison.

Firstly, we know that for honest experts, the standard rule performs close to optimally. For every $\delta > 0$ we can pick a learning rate $\eta$ such that as $T \to \infty$

the loss of the algorithm $M^{(T)} \leq (2+\delta)m_i^{(t)}$, while no algorithm could do better than $M^{(T)} < 2m_i^{(T)}$ [Littlestone and Warmuth, 1994, Freund and Schapire, 1997]. This motivates looking at strictly proper scoring rule that are "close" to the standard update rule in some sense. In Figure 5.3, if we compare the two strictly proper scoring rules, the spherical rule seems to follow the standard rule better than Brier does.

A more formal way of look at this is to look at the scoring rule gap. In Figure 5.3 we marked the $p = \frac{1}{2}$ location. Visually, the scoring rule gap $\gamma$ is the difference between a scoring rule and the standard rule at $p = \frac{1}{2}$. Since the Brier score has a large scoring rule gap, we're able to prove a strictly stronger lower bound for it: the scoring rule gap $\gamma = \frac{1}{4}$, hence MW with the Brier scoring rule cannot do better than $M^{(T)} \geq (2+\frac{1}{4})m_i^{(T)}$ in the worst case, according to Lemma 5.20. Corollary 5.21 shows that for the Spherical rule, this factor is $2+\frac{1}{5}$. The ability to prove stronger lower bounds for scoring rules with larger gap parameter $\gamma$ is an indication that it is probably harder to prove strong upper bounds for those scoring rules.

# Chapter 6

# Incentives in Bitcoin Mining Pools

In this chapter we continue our approach to model game-theoretical aspects of applications that have lacked analysis of incentives for participants.[1] We introduce a game-theoretic model for reward functions within a single Bitcoin mining pool. Our model consists only of an unordered history of reported shares and gives participating miners the strategy choices of either reporting or delaying when they discover a share or full solution. We defined a precise condition for incentive compatibility to ensure miners strategy choices optimize the welfare of the pool as a whole. With this definition we show that proportional mining rewards are not incentive compatible in this model. We introduce and analyze a novel reward function which is incentive compatible in this model. Finally we show that the popular reward function pay-per-last-N-shares is also incentive compatible in a more general model.

## 6.1 Introduction

By almost any measure, Bitcoin [Nakamoto, 2008] has become the most successful cryptocurrency in history. While Bitcoin has evolved into a very complex sociotechnical system which we will not describe in detail here,[2] at its core lies a decentralized consensus protocol allowing all participants to agree on a common global ledger of transactions to prevent double-spends and other disallowed behavior. The key to Bitcoin's consensus protocol (sometimes more broadly called *Nakamoto consensus* after its founder) is a group of entities called *miners* who race to solve a challenging cryptographic puzzle for the right to append a new block of transactions to Bitcoin's ledger, the *blockchain*. A system of incentives encourages these miners to follow the protocol faithfully in exchange for the ability to earn newly-minted coins and transaction fees

[1] This chapter is based on joint work with Joe Bonneau, Dan Boneh and Tim Roughgarden and was presented at Financial Crypto '16 [Schrijvers et al., 2016]

[2] For an academic overview of Bitcoin we refer the reader to [Bonneau et al., 2015].

in proportion to the amount of computational effort they have expended (also called hashing power or mining power).

Finding a single Bitcoin block is very rewarding (today worth at least Ƀ25, over US$6,000), yet it is also very difficult for smaller miners who might find a block on expectation only every few months or even every few years. As a result, the majority of mining power now consists of miners participate in *mining pools* in which they agree to divide rewards from blocks found by any member of the pool and thus receive a steadier stream of income. Choosing the exact algorithm used to divide up mining pool rewards (the *reward function*) however, turns out to be a challenging incentive design problem.

Pools are sometimes controversial in the Bitcoin community as they represent a form of centralization. Miller et al. [2014] proposed a future cryptocurrency which attempts to prevent their formation. As of today though they are an indispensable part of Bitcoin as well as many related cryptocurrencies (to which our analysis also applies).

Despite their importance to the Bitcoin ecosystem, relatively little work has analyzed on the reward functions underlying pools. Rosenfeld provided an initial overview of the space [Rosenfeld, 2011] and introduced *pool-hopping* attacks, whereby miners switch between pools to maximize profit. Lewenberg et al. [2015] showed that in certain circumstances no reward function can prevent all pool-hopping. Several authors have studied *withholding attacks* between pools, whereby pools infiltrate competing pools, collecting rewards but withholding valid shares to damage their competitors [Eyal, 2015, Courtois and Bahack, 2014, Luu et al., 2015]. Counter-intuitively, this attack can be profitable in some plausible circumstances. Other work has focused on pools more directly attacking each other via denial-of-service attacks on the network [Laszka et al., 2015, Johnson et al., 2014].

In this paper we introduce a formal game-theoretical framework to study the reward functions for a *single* pool in which participants can choose when to report valid shares can but cannot change pools or solo mine. To the best of our knowledge, ours is the first treatment of this model and the attacks we describe are novel. We are motivated by a very natural question: if individual miners are interested in maximizing their expected utility, is their behavior optimal for the pool as a group? For example, if miners are incentivized to delay reporting full solutions to the pool, this may lower the pool's overall rewards and even make it more vulnerable to external sabotage [Eyal, 2015]. Although our single-pool model is deliberately simplified, we still show that some reward functions used in practice such as simple proportional payments are not incentive compatible.

We introduce a novel reward function which is incentive compatible within our model while still maintaining other desirable properties. Our reward function will remain incentive compatible even in a more complex informational model (although

it may need to be extended if the definition of incentive compatibility is extended to include more complicated attacks).

While our model cannot capture all reward functions used in practice, we consider it an important new model in analyzing mining pool reward functions in Bitcoin and related systems. We further take the first step to analyzing reward functions in more general informational models by carefully examining the popular pay-per-last-N-shares reward function, and show that it is incentive compatible. This indicates that our approach is not limited to the informational assumptions, but can be more generally applied.

## 6.2 Preliminaries

In this paper we look at a simple model in which miners are bound to working for a particular pool and where their strategic choice is the following: if a miner finds a solution to the cryptographic puzzle, *when* does it report this to the pool. The pool is run by a *pool operator* and contains a fixed number $n$ of miners. Each miner $i$ has a fraction $\alpha_i$ of the total mining power. For most of this paper we will assume that $\sum_{i=1}^{n} \alpha_i = 1$, meaning that the pool has all the available mining power; there are no other pools or solo miners. In Appendix 6.C we look at the case where the pools total hashing power $\alpha_P = \sum_{i=1}^{n} \alpha_i < 1$ and show that while this makes a quantitative difference, qualitatively our results carry over.

The time it takes for a miner to find a share is an exponentially distributed random variable with parameter $\alpha_i$; hence in expectation it takes time $1/\alpha_i$ to find a share. Each share is also a full solution with probability $1/D$.

### 6.2.1 Reward Functions and History Transcripts

Miners report their shares and solutions to the pool operator. When a solution is reported, the operator who collects the block reward from the Bitcoin network and subsequently divides the reward among the $n$ miners according to a *reward function* $R$. The game then restarts. For the mathematical model we assume no variability in the block reward or the transaction fees, although the work can be extended to include this.

The reward function is the only way in which the miners receive any payout and therefore the reward function completely drives the behavior of miners. A perfectly equitable reward function would simply give each miner $i$ a fraction $\frac{\alpha_i}{\alpha_P}$ of the reward in proportion to the fraction of the pool's total mining power to which that miner contributed.

However, the pool operator does not know the actual $\alpha_i$ of each miner. The challenge in designing a reward function $R$ stems from the necessity of estimating this

based on reported shares and solutions. The operator's ability to estimate $\alpha_i$ depends on the precise information it has access to. We model this as a *history transcript* $\mathcal{H}$. A reward function $R : \mathcal{H} \to [0,1]^n$ is a function from a history transcript to an allocation $\{a_i\}_{i=1}^n$ with $\sum_i a_i = 1$. We use $R_i : \mathcal{H} \to [0,1]$ to denote the function that yields the $i^{\text{th}}$ component of $R$.

In most of this paper we analyze the case of an unordered history transcript:[3] $\mathcal{H}$ contains for each miner $i$ the total number of shares $b_i \in \mathbb{N}$ that have been reported in that round.[4] Thus, the history transcript is given by a vector $\mathbf{b} \in \mathbb{N}^n$ that contains for each of the $n$ players the total number of shares that she found during the round (where the full solution is also counted as a share). We use vector notation for $\mathbf{b}$, so $\mathbf{b}_1 + \mathbf{b}_2$ means the component-wise addition of $\mathbf{b}_1$ and $\mathbf{b}_2$, and $||\mathbf{b}||_1 = \sum_{i=1}^n b_i$ is the sum of the components of $\mathbf{b}$.

This model is perhaps the simplest possible[5] which enables a mining pool to function, yet it captures several basic reward function schemes used in practice. There are also reward functions which require additional information, such as the order in which shares were reported or reports from previous rounds of the game. In Section 6.8 we briefly discuss how to generalize this model and the challenges with characterizing incentive compatibility for them. However, we stress that positive results demonstrated incentive compatibility in our simple model extend to any more complicated model, as the pool operator can always decline to use additional information in its reward function.

### 6.2.2 Miner Strategy

Now that we defined a model and the reward function $R$, let's look at how the choice of $R$ impacts the behavior of miners. The goal of any mining pool is to earn as many rewards as possible for its members.[6] If miners delay in reporting blocks to the pool, this imposes a risk that an external pool may find the block first, undermining the pool's potential rewards.[7] Note that while we are only modeling a single pool, we build in the assumption that this pool wants to report solutions as fast as possible to

[3] In Section 6.6 we consider a strictly more general informational model, which will be described there.

[4] We adopt the convention that $\mathbb{N}$ includes the number 0.

[5] A simpler format such as only receiving information about which miner reported a full solution would only allow a replication of solo mining.

[6] In this work we are only considering a pool which follows the default mining strategy and does not attempt to implement an deviant strategies to earn disproportionately more rewards than competing pools, such as temporary block withholding [Eyal and Sirer, 2014].

[7] Another way of saying this is that a reward function which does not compel participants to report solutions immediately is *not* welfare maximizing, since the selfish behavior of individuals can hurt the total reward of the group.

the wider network to avoid getting scooped by the competition. Thus we will want our reward function to ensure solutions to be reported and processed as soon as they are found.[8]

In response to a reward rule $R$, miners choose a strategy $\sigma(R)$ which dictates what a miner does when it finds a share or full solution. Ideally the strategy $\sigma(R)$ is to report any share or solution immediately. However, the pool operator cannot directly tell miners what to do; rather they should choose an $R$ such that the miners corresponding strategy $\sigma(R)$ is to immediately report. Formally, let $t$ be the time since miner $i$ started mining, let $T$ be the number of rounds that have been completed at time $t$, and let $\mathbf{b}_j$ be the number of shares per player in round $j$. Miner $i$ is interested in maximizing their throughput:

$$\sigma(R) := \max_{\sigma} \lim_{t\to\infty} \frac{\sum_{j=1}^{T} R_i(\mathbf{b}_j)}{t}. \tag{6.1}$$

Here $\sigma$ impacts the number $T$ of rounds that were completed, as well as the number of reported shares $\mathbf{b}_j$ in each round.

### 6.2.3 Reward Function Desiderata

We define three properties which are important for a reward function. The first is a formalization of the intuition above:

**Property 6.1** (Incentive Compatibility)**.** *A reward function $R$ is* incentive compatible *when every miner's best response strategy $\sigma(R)$ reports full solutions immediately.*

In Section 6.3 we give a mathematical condition that characterizes Property 6.1, and that can easily be verified for reward functions.

Next, we require that the pool pays miners in proportion to the amount of work they have performed. Miners form pools to reduce the variance in revenue. In practice they might accept losing a small fraction $f$ of their expected value in fees, but we would like miner performing an $\alpha_i$ fraction of the work to receive an $\alpha_i$ fraction of the reward.

**Property 6.2** (Proportional Payments)**.** *A reward function $R$ provides* proportional payments *whenever for each miner $i$*

$$\mathbb{E}_{\mathbf{b}}[R_i(\mathbf{b})] = \alpha_i.$$

[8]While we do not consider fees in this paper, note that a pool operator would also want to optimize throughput if collects a fraction of the reward.

Finally, we would like the pool operator to never incur a deficit. That is, the reward function $R$ should precisely divide the reward among the $n$ miners at the end of a round. If this is not the case, then either miners may leave some value on the table, or the pool operator may be liable for more then she received herself. This latter condition is particularly dangerous, as it leaves the pool exposed to sabotage attacks [Eyal, 2015] in which competing miners purposely withhold full solutions to damage the pool.

**Property 6.3** (Budget Balanced)**.** *A reward function $R$ is $(\gamma, \delta)$-budget balanced when for all $\mathbf{b}$:*

$$\gamma \leq \sum_{i=1}^{n} R_i(\mathbf{b}) \leq \delta.$$

In particular, an $(\gamma, 1)$-budget balanced reward function will never pay more to the miners than the pool operator received. Our goal will be $(1, 1)$-budget balanced reward functions which share the reward exactly among the $n$ miners.

### 6.2.4 Common examples

Perhaps the most obvious reward function is the *proportional* reward function: $R_i(\mathbf{b}) = \mathbf{b}_i/K$, where $K = ||\mathbf{b}||_1 = \sum_{i=1}^{n} b_i$. That is, the reward is shared proportional to the number of shares each miner reported. We show that the proportional rule is not incentive compatible in Section 6.4.1.

Another reward function is the *pay-per-share* reward function: $R_i(\mathbf{b}, s) = b_i/D$, where participants are rewarded a fixed amount per share. In Section 6.4.2 we show that while this method is incentive compatible, it is not budget balanced (defined in Section 6.2.3), which means that the pool operator may be liable to pay out more to miners than she collects from the Bitcoin protocol.

### 6.2.5 Ensuring Steady Rewards

Miners are interested in maximizing the total reward they receive per time unit, but they join pools primarily to achieve a more consistent stream of revenue. Our goal will be to build a reward function which is as consistent as possible which satisfies the three properties above. It is tempting to isolate one metric, such as the variance or standard deviation of the distribution of rewards, but we will discuss in Section 6.7 why these metrics are probably not the best measures of consistency in practice and provide different simulation results to compare reward functions.

## 6.3 Incentive Compatibility

We stated that for a reward function $R$ to be incentive compatible, it needs to incentivize miners to report full solutions immediately. In this section we express that as a condition that can easily be checked for any given reward function.

We do this by looking at the strategic choice that a miner faces when she finds a full solution. Either she reports the full solution immediately, or she decides to delay reporting the solution until $d$ more shares have been found. In this section we do not take into account the possibility of another miner finding and reporting a solution during this delay. In Appendix 6.B we show that this is virtually without loss of generality.

Consider the situation when at time $t$, miner $i$ finds a full solution. At this point $\mathbf{b}_t$ shares have been reported to the pool operator (for notational simplicity we assume that the full solution is already included in $\mathbf{b}_t$). The action space of miner $i$ is $d \in \mathbb{N}$ (including 0) where the miner waits for $d$ additional shares to be reported before reporting the full solution. If miner $i$ decides to wait for $d$ shares before reporting, her expected reward at the end of the delay is:

$$\mathbb{E}_{(\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d)}[R_i(\mathbf{b}_t + \mathbf{b})] = \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d} \Pr(\text{seeing } \mathbf{b}) \cdot R_i(\mathbf{b}_t + \mathbf{b})$$

On the other hand, if she decides to report the full solution immediately, she will receive the reward she was entitled to at that moment and a new round will start. So she will additionally get $d$ times the expected reward per share. That is, delaying her report imposes an opportunity cost by not beginning the next round. So her expected reward in this situation after $d$ more shares is:

$$\begin{aligned} R_i(\mathbf{b}_t) + d \cdot \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{\mathbb{E}_{\mathbf{b}}[||\mathbf{b}||_1]} &= R_i(\mathbf{b}) + d \cdot \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{\sum_{k=1}^{\infty} k \left(1 - \frac{1}{D}\right)^{k-1} \frac{1}{D}} \\ &= R_i(\mathbf{b}_t) + \frac{d}{D} \cdot \mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \end{aligned}$$

Reporting the solution immediately will be more profitable than delaying for $d$ shares if and only if:

$$\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)) \;\leq\; \frac{d}{D} \cdot \mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \tag{6.2}$$

The miner's best strategy is to report immediately if this condition holds for all $d \in \mathbb{N}\backslash\{0\}$. The following lemma states that there exists a $d \in \mathbb{N}\backslash\{0\}$ for which this condition holds if and only if it holds for $d = 1$. This is a very powerful statement: to determine the incentive compatibility of a reward function, we only need to see if it is profitable to delay reporting for a single additional share. In the following, let $\mathbf{e}_j$ be the $j^{\text{th}}$ standard basis vector that is 0 everywhere except for the $j^{\text{th}}$ component which is 1. The proof appears in Appendix 6.A.

**Lemma 6.4.** *For a reward function $R$, a player $i$ has an incentive to report full solutions immediately, iff the following condition holds for all $\{\alpha_i\}_{i=1}^n, \mathbf{b}_t, D, i$:*

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) \;\leq\; \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{D} \tag{6.3}$$

So to show that a reward function is incentive compatible, we need to show that condition (6.3) holds, and conversely when we show that for a reward function condition (6.3) is not guaranteed to hold, it cannot be incentive compatible.[9]

While for incentive compatibility we do not care about miners reporting shares immediately, this is important in ensuring proportional payments.

**Lemma 6.5.** *Miners report shares immediately if and only if the reward function $R$ is monotonically increasing each component. That is: for all $i$, and $\mathbf{b}$:*

$$R_i(\mathbf{b} + \mathbf{e}_i) > R_i(\mathbf{b}).$$

*Proof.* Since the order or timing of shares does not matter, for analysis purposes we can assume the following scheme: as soon as a full solution is reported the pool operator asks all miners for the shares that they found. If the reward function $R$ is monotonically increasing then each additional share that $i$ reports increases her share, hence she will report all shares. Conversely, if $R$ is not monotonically increasing at some $\mathbf{b}$, then if miner $i$ has $b_i + 1$ shares, and all other miners have reported shares according to $\mathbf{b}$, then she will not report her last share.

Now consider the original problem: when a miner finds a share, will she report it immediately? If she finds a share and the reward function is monotonically increasing, then reporting it immediately can only increase her reward, whereas delaying it may mean that someone else reports the full solution before she reports her share, in which case she loses the opportunity to report. Thus she will report immediately. □

[9] In Appendix 6.B we show that the possibility of another miner reporting a solution does not materially change the characterization here and in Appendix 6.C we extend this to include the possibility of another pool reporting a full solution.

## 6.4 Incentive Compatibility of Existing Methods

Now we will apply our characterization of incentive compatibility to reward functions which are in use today. In this section we restrict ourselves to reward functions that can be modeled by our definition of history transcript as described in Section 6.2.

### 6.4.1 Proportional Reward Function $R^{(\mathbf{prop})}$

One of the earliest reward functions that is still in use is the proportional reward function. The idea is to divide the reward according to the proportion of shares of a miner compared to all shares that were reported to the pool:

$$R_i^{(\text{prop})}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}.$$

In expectation, the reward per share for each player is $\alpha_i/D$. This approach is clearly proportional and budget-balanced. Previous work [Rosenfeld, 2011] has shown that in the presence of multiple pools, miners can be incentivized to change pools after a certain number of shares has been found. In this section we present a new problem that exists even in the absence of other mining pools and means that the proportional reward function is not incentive compatible. The proof appears in Appendix 6.A

**Lemma 6.6.** *The proportional rule* $R_i^{(prop)}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}$ *is not incentive compatible.*

This result shows that the proportional reward function is *not* incentive compatible for a fundamental reason distinct from previous criticism. Even in the absence of other pools, it does not always compel miners to report solutions immediately. The intuition behind Lemma 6.6 is that if a player discovers a full solution early but has been unlucky and reported a lower number of shares than they would expect based on their mining power, it is in their incentive to delay reporting their solution since on expectation their fraction of all reported shares will go up. We can draw a number of corollaries immediately from Lemma 6.6:

- If the current ratio of blocks exceeds the expected ratio, then a player $i$ would report any full solution immediately.
- If the current ratio of blocks is *lower* than a player's expected ratio, then she might hold off to make up for this discrepancy.
- With fewer shares found, it's easier for a player to catch up, hence she is more willing to hold off reporting.
- After $D$ shares have been found, any player will always report a full solution immediately, even if she has not found a single share herself.

### 6.4.2 Pay-Per-Share Reward Function $R^{(\text{pps})}$

The pay-per-share reward function pays a fixed amount for every share that is reported. Recall that each share is a full solution with probability $1/D$, in expectation the pool operator sees $D$ shares for every full solution. Therefore the payout per share is $1/D$ leading to the following reward function:

$$R^{(\text{pps})}(\mathbf{b}) = \frac{b_i}{D}.$$

It's easy to see that pay-per-share *is* incentive compatible.

**Lemma 6.7.** *The pay-per-share rule $R_i^{(pps)}(\mathbf{b}) = \frac{b_i}{D}$ is incentive compatible.*

*Proof.* The left hand side of (6.3) evaluates to

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left( R_i^{(\text{pps})}(\mathbf{b}_t + \mathbf{e}_j) - R_i^{(\text{pps})}(\mathbf{b}_t) \right) \\
&= \alpha_i \cdot \left( \frac{b_i + 1 - b_i}{D} \right) + (1 - \alpha_i) \cdot \left( \frac{b_i - b_i}{D} \right) \\
&= \frac{\alpha_i}{D}
\end{aligned}$$

and the right hand side evaluates to

$$\frac{\mathbb{E}_{\mathbf{b}} \left[ R_i^{(\text{pps})}(\mathbf{b}) \right]}{D} = \frac{\alpha_i}{D}.$$

□

This result comes at no surprise: with pay-per-share there is no benefit to delay reporting a full solution as you receive a constant payment for every reported share (or full solution). As discussed before though, it is not budget balanced.

**Proposition 6.8.** *The pay-per-share rule $R_i^{(pps)}(\mathbf{b}) = \frac{b_i}{D}$ is no better than $(1/D, \infty)$-budget balanced.*

*Proof.* On one extreme, if a full solution is reported before any other shares have been reported, then $R^{(\text{pps})}$ pays out $\sum_{i=1}^{n} R_i^{(\text{pps})}(\mathbf{b}) = 1/D$. On the other extreme there is no bound on how many shares can be found before a full solution must be obtained. Hence $R^{(\text{pps})}$ cannot be $(1/D, C)$-budget balanced for any finite $C$. Therefore it is $(1/D, \infty)$-budget balanced. □

In the absence of sabotage attacks, with the pay-per-share rule the pool operator pays out no more than it takes *in expectation*, but keeping the probability of bankruptcy low requires large reserves for the pool operator [Rosenfeld, 2011]. There are several variations that ameliorate the budget balance problem [Rosenfeld, 2011, Sec 4], none of them quite satisfactorily.

## 6.5 A New IC Reward Function

In the last section we saw that two common existing methods which are possible in our model both lack one of the desiderata for reward functions. The proportional reward function may incentivize miners to delay reporting of solutions, whereas the pay-per-share function may make the pool operator liable for more than she receives from the protocol. In this section we demonstrate a reward function that satisfies all three desiderata of reward functions, while still guaranteeing a steady stream of rewards for all participants.

To satisfy the proportional payments property, it is necessary to estimate the proportion of work that each miner has done. The only information the pool operator receives within our informational model is the total number of shares per miner in the round. When only a few shares have been found in the round, every additional share may change this estimation quite significantly. When satisfying the budget-balanced property, this must translate into a large change in the payout. When there is the possibility of a large payout for an extra share, this may lead to incentive compatibility issues. Note that in practice pay-per-share reward schemes usually avoid this problem by lowering the payment amount in these cases. So to give a scheme that meets all three desiderata, we need to take an additional estimator for $\alpha_i$ into account. In the next subsection we show that we can use the identity of the discoverer of the full solution as this estimator.

### 6.5.1 The IC Reward Function

We propose the reward function $R_i^{(\text{ic})} : \mathbb{N}^n \times \{1, ..., n\} \to [0, 1]$, that in addition to a count of the shares per miner also includes the identity of the discoverer of the full solution. In the following let $\mathbb{1}\{c\}$ be the indicator function that is 1 if $c$ is true, and 0 otherwise.

$$R_i^{(\text{ic})}(\mathbf{b}, s) = \frac{b_i}{\max\{||\mathbf{b}||_1, D\}} + \mathbb{1}\{i = s\} \cdot \left(1 - \frac{||\mathbf{b}||_1}{\max\{||\mathbf{b}||_1, D\}}\right)$$

There are two cases to consider for the reward function. The easiest is when the

total number of reported shares $||\mathbf{b}||_1 \geq D$. In that case $\left(1 - \frac{||\mathbf{b}||_1}{\max\{||\mathbf{b}||_1, D\}}\right) = 0$, hence the reward function is identical to the proportional function. When $||\mathbf{b}||_1 < D$ each share receives a fixed reward of $1/D$, like in the pay-per-share function. However, this would leave some money on the table as the total payout would be $||\mathbf{b}||_1/D$ and $||\mathbf{b}||_1 < D$. So the remainder of the reward is given to the discoverer of the full solution.

**Lemma 6.9.** *The reward function $R^{(ic)}$ provides proportional payments.*

*Proof.* We first split the expression into the two relevant cases.

$$\begin{aligned}\mathbb{E}_{\mathbf{b}}\left[R_i^{(\mathrm{ic})}(\mathbf{b}, s)\right] = \Pr(||\mathbf{b}||_1 < D) \cdot \mathbb{E}_{\mathbf{b}}\left[R_i^{(\mathrm{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 < D\right] \\ + \Pr(||\mathbf{b}||_1 \geq D) \cdot \mathbb{E}_{\mathbf{b}}\left[R_i^{(\mathrm{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 \geq D\right].\end{aligned}$$

We now show that in both cases the expected reward for miner $i$ is $\alpha_i$. When $||\mathbf{b}||_1 \geq D$ the IC rule is no different than the proportional rule, hence

$$\mathbb{E}_{\mathbf{b}}\left[R_i^{(\mathrm{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 \geq D\right] = \alpha_i.$$

Now for the case where $||\mathbf{b}||_1 < D$:

$$\begin{aligned}&\mathbb{E}_{\mathbf{b}}\left[R_i^{(\mathrm{ic})}(\mathbf{b}, s) \mid ||\mathbf{b}||_1 < D\right] \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k \mid ||\mathbf{b}||_1 < D) \cdot \left(\frac{\mathbb{E}[b_i \mid ||\mathbf{b}||_1 = k]}{D} + \Pr(i = s) \cdot \left(1 - \frac{k}{D}\right)\right) \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k \mid ||\mathbf{b}||_1 < D) \cdot \left(\frac{\alpha_i \cdot k}{D} + \alpha_i \cdot \left(1 - \frac{k}{D}\right)\right) \\ &= \sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k \mid ||\mathbf{b}||_1 < D) \cdot \alpha_i \\ &= \alpha_i\end{aligned}$$

Here we used the fact that when a full solution is found, the probability that it was discovered by miner $j$ is exactly its power $\alpha_j$. □

**Theorem 6.10.** *The reward function $R^{(ic)}$ is incentive compatible.*

*Proof.* By Lemma 6.9, the right-hand side of condition (6.3) is $\alpha_i/D$. Now for the left-hand side; if $||\mathbf{b}||_1 \geq D$ the rule is identical to $R^{(\mathrm{prop})}$, so the left-hand side is at

most $\alpha_i/D$, hence condition (6.3) holds in this case. So the only case left to prove is when $||\mathbf{b}||_1 < D$.

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left( R_i^{(\text{ic})}(\mathbf{b}_t + \mathbf{e}_j, i) - R_i^{(\text{ic})}(\mathbf{b}_t, i) \right) \\
&= \alpha_i \cdot \left( \frac{b_i + 1 - b_i}{D} \right) + (1 - \alpha_i) \cdot \left( \frac{b_i - b_i}{D} \right) + \left( 1 - \frac{||\mathbf{b}||_1 + 1}{D} \right) - \left( 1 - \frac{||\mathbf{b}||_1}{D} \right) \\
&= \frac{\alpha_i}{D} - \left( \frac{||\mathbf{b}||_1 + 1}{D} - \frac{||\mathbf{b}||_1}{D} \right) \\
&= \frac{\alpha_i - 1}{D} \\
&\leq 0
\end{aligned}$$

So when $||\mathbf{b}||_1 < D$ the miner is expected to *lose* utility by delaying, and certainly condition (6.3) holds. So in all cases condition (6.3) holds, hence by Lemma 6.4 $R^{(\text{ic})}$ is incentive compatible. □

So $R^{(\text{ic})}$ satisfies proportional payments and incentive compatibility. Finally, it is also a $(1, 1)$-budget balance reward function.

**Proposition 6.11.** *$R^{(ic)}$ is a $(1, 1)$-budget balanced reward function.*

*Proof.* When $||\mathbf{b}||_1 < D$ the total payout is $\sum_{i=1} \frac{b_i}{D} + \sum_{i=1} \frac{b_i}{D} + \frac{D - ||\mathbf{b}||_1}{D} = \frac{||\mathbf{b}||_1}{D} + \frac{D - ||\mathbf{b}||_1}{D} = 1$. When $||\mathbf{b}||_1 \geq D$ the total payout is $\sum_{i=1} \frac{b_i}{||\mathbf{b}||_1} = \frac{||\mathbf{b}||_1}{||\mathbf{b}||_1} = 1$. □

### 6.5.2 Providing a Steady Payment Stream

While $R^{(\text{ic})}$ satisfies all three desiderata for reward functions, it might be a concern that the reward function pays out a potentially large fraction of the reward to a single miner. Miners join a pool because they prefer a steady stream of small payments over periodic large payments. We show here that the majority of the reward is paid out for shares and not full solutions, and hence that the majority of the pool's rewards are redistributed in a steady stream. This ameliorates a large part of the problem of mining alone, while guaranteeing incentive compatibility. In the Section 6.7 we give simulation results suggesting that this payment stream is sufficiently steady in practice.

**Lemma 6.12.** *In expectation, a fraction $1 - e^{-1} \approx 0.63$ of the rewards are given based on shares under $R^{(ic)}$.*

*Proof.* The fraction of the reward given to the discoverer of the full solution is

$$\begin{aligned}\sum_{k=1}^{D-1} \Pr(||\mathbf{b}||_1 = k) \cdot \left(1 - \frac{k}{D}\right) &= \sum_{k=1}^{D-1} \frac{1}{D} \cdot \left(1 - \frac{1}{D}\right)^{k-1} \cdot \left(1 - \frac{k}{D}\right) \\ &= \left(1 - \frac{1}{D}\right)^{D} \\ &\leq e^{-1}\end{aligned}$$

The remainder of the reward is split among the reported shares, hence the payout to shares in expectation is $1 - e^{-1} \approx 0.63$. □

## 6.6 Incentive Compatibility of PPLNS

In previous sections we have given an overview of incentive compatibility for any reward function based on access to a history transcript $\mathcal{H}$ consisting of a count of all reported shares. By deriving incentive compatibility at this high level of abstraction allowed us to easily prove incentive compatibility for any function within this informational model.

In this section we look at incentive compatibility of a *particular* reward function that require a *more general* informational model: the Pay-Per-Last-$N$-Shares (PPLNS) reward function, that is widely used in practice. We first discuss the required changes in the informational model, and how the PPLNS function works, and then we show that the function is incentive compatible.

### 6.6.1 The PPLNS Reward Function

The PPLNS reward function $R^{(\text{pplns})}$ differs from the reward functions seen so far in two important ways. Firstly, it maintains a history of reported shares that spans *multiple* rounds. So what happens in round $T$ is no longer isolated from what happens in round $T+1$. Secondly, the method takes the order of reported shares into account in a specific way: it maintains a sliding window of length $N$ and divides the reward proportionally over these $N$ shares. So the history transcript $\mathcal{H}$ that $R^{(\text{pplns})}$ uses is $\mathbf{s} = [s_{t-N}, s_{t+1-N}, ..., s_t]$ (an ordered list of $N$ elements) and the reward function is:

$$R^{(\text{pplns})}(\mathbf{s}) = \frac{\#\{s_j : s_j \in \mathbf{s} \wedge s_j = i\}}{N}$$

Since the order of reports matter, we say that shares fall into *slots*. Each slot states if the report contains either a full solution or a share, and who the miner was that reported it.

### 6.6.2 Incentive Compatibility of PPLNS

We do not have a general condition under which reward functions in this informational model are incentive compatible, so we argue incentive compatibility directly. In this section we consider both reporting of shares as well as full solutions. For each, we consider a binary strategy space: either report the share/solution immediately, or delay reporting until one more share is found.[10]

**Lemma 6.13.** *For the reward function $R^{(pplns)}$; a miner $i$ reports shares immediately when her mining power $\alpha_i < 1 - \frac{D}{N}$.*

*Proof.* We directly calculate the expected revenue for delay versus reporting. When the miner decides to delay reporting a share until one more share/solution is found, she aims to move the sliding window of slots for which the share is eligible to receive reward one further into the future. This means that –as long as no other miner finds a full solution and reports it– the share is active for $N - 1$ of the same slots, so any reward she receives from full solutions in those slots she will get regardless of her choice to report immediately versus delaying. On the upshot, it could be the case that the one additional slot she's eligible for in the future yields a full solution. This will happen with probability $1/D$ (since a share constitutes a full solution with probability $1/D$) and in that case the share gets an extra payout of $1/N$ for the delayed share, yielding an expected benefit for delaying of $1/ND$.

However, there is also a risk associated with delaying. With probability $1 - \alpha_i$ a share will be found by a different miner, and with probability $1/D$ it will constitute a full solution. When this happens, miner $i$ will no longer be able to report the share as it was discovered for a previous round. The expected value per share is $1/D$ (as it's active for $N$ rounds, in which in expectation $N/D$ full solutions will be reported for a value of $1/N$ each) hence the expected harm for delaying the report is $(1 - \alpha_i)\frac{1}{D^2}$.

So the miner will report the share immediately iff $\frac{1}{ND} < (1 - \alpha_i) \cdot \frac{1}{D^2}$. Plugging in $\alpha_i < \frac{D}{N}$ leads to $(1 - \alpha_i) \cdot \frac{1}{D^2} \geq \frac{D}{N} \cdot \frac{1}{D^2} = \frac{1}{ND}$ so the condition holds, and miners report shares immediately. □

In the previous section there were potential benefits, and harms to delaying the report of a share, but there was no opportunity cost. Since miner $i$ did not have a full solution, her delay did not cause unnecessary work for all miners in the pool. For full solutions we have to take the opportunity cost of letting all miners work on a block for which a full solution is already found into account. This will guarantee incentive compatibility whenever $N > D$.

**Lemma 6.14.** *For the reward function $R^{(pplns)}$; a miner $i$ reports full solutions immediately when $N \geq D$.*

[10] This makes our results slightly less general in this setting than for the reduced information setting, where the miner could delay for any delay $d$.

*Proof.* In delaying a full solution, the hope is to get another share to report before the miner reports the full solution. This happens with probability $\alpha_1 \cdot \frac{D-1}{D}$ (we need the share to *not* be a full solution) and the additional value to this share would be $\frac{1}{N}$ compared to it being reported after the full solution. However, while waiting for a share, with probability $\frac{1}{D}$ the next share will be a full solution, either found by miner $i$, or one of the other miners. Regardless of who finds the solution, the previous full solution that miner $i$ was sitting on has become worthless: either a different miner reported the full solution ending the round and thus making the delayed full solution worthless, or miner $i$ now has 2 full solutions of which she can report only one. When this happens, she loses the solution whose expected value is $1/D$ (as this is counted as a share for future). So the expected upshot for delaying the solution is $\alpha_1 \frac{D-1}{D} \frac{1}{N}$ and the expected harm is $\frac{1}{D^2}$.

In addition to this, when the miner chooses to delay until one more share is found, she lets all miners in the pool work on a block for which she already has a solution. If everyone were to spend that effort on a new block, that work would in expectation constitute $1/D$ of the work for a new block, of which in expectation miner $i$ would receive $\alpha_i$ of the reward. Thus, the opportunity cost is $\frac{\alpha_i}{D}$. Therefore, a miner will report a full solution immediately iff $\alpha_i \frac{D-1}{D} \frac{1}{N} - \frac{1}{D^2} \leq \frac{\alpha_i}{D}$ which holds whenever $N \geq D$. □

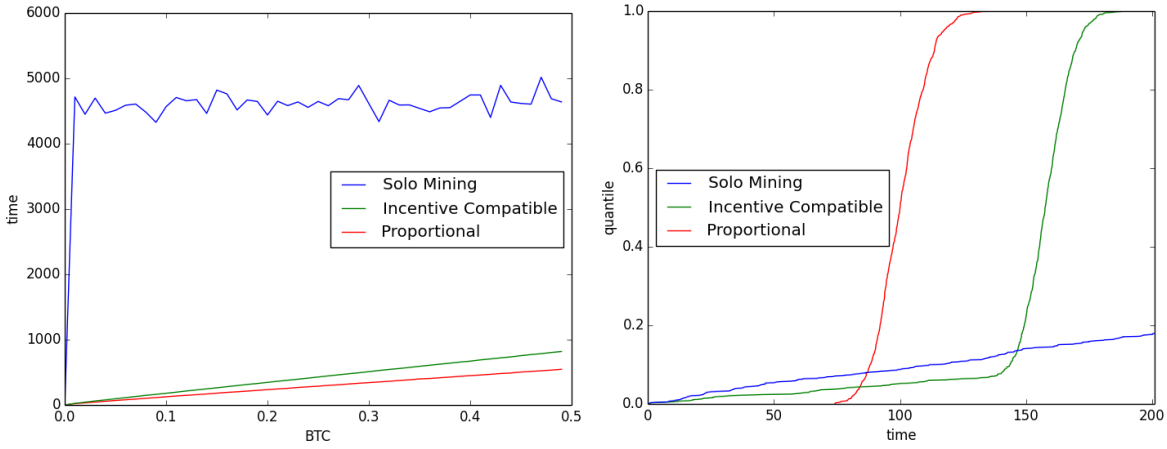
## 6.7 Simulations

The typical way to compare different reward function is to look at the variance of payout for a single share [Rosenfeld, 2011], with a lower variance considered better. However, looking at the variance alone may lead to some undesirable conclusions. Consider the following two examples.

**Example 6.15** (Solo Mining)**.** *When solo mining, the reward of a share is 1 if it constitutes a full solution, and 0 otherwise. The variance is* $\mathbb{E}_{\mathbf{b}}[R_i^{(solo)}(\mathbf{b})^2] - \mathbb{E}_{\mathbf{b}}[R_i^{(solo)}(\mathbf{b})]^2 = \frac{1}{D} - \frac{1}{D^2} = \frac{D-1}{D^2} \approx \frac{1}{D}$*. This value* decreases *with $D$. However, a solo miner is completely indifferent as to how many shares are typically found before a full solution is discovered. She only cares about the full solutions.*

**Example 6.16** (Pay-per-Share)**.** *Consider the pay-per-share reward function, where the miner receives $1/D$ per share she reports. The variance of pay-per-share is 0, since the miner always receives precisely the same amount whenever she reports a share. However, when $D$ gets smaller, the time between payouts increases, and in the extreme case of $D = 1$, the scheme is identical to solo mining, while the variance is still 0.*

In these two example we've seen a reward rule whose utility should not change with the parameter $D$, but does, and one that should change with the parameter $D$, but doesn't. The variance of payments for a share addresses the uncertainty in payouts, but it falls short in describing the utility that miners get out of the reward. What a miner is interested in when joining is pool, is that with high probability, it wants a guarantee on some minimal payout.[11]




(a) $99^{\text{th}}$ percentile time to earn rewards

(b) CDF of time to earn ₿ 0.1

Figure 6.1: Simulation results for our new incentive-compatible reward function.

In Figure 6.1a we plot the time it takes for a miner to gain a given number of bitcoins with 99% certainty. We run a simulation for a miner $i$ with $\alpha_i = 0.001$, $D = 1,000,000$. A unit of time corresponds to the expected time it takes for all miners combined to find a full solution (in reality this is about 10 minutes) and we normalize the reward for finding a full solution to be ₿ 1 (in reality this currently about ₿ 25, although it changes over time). The lines indicate for each of three reward functions how long one has to wait to gain a given amount of ₿ with 99% probability. First of all, observe that for solo mining the time is about 4500 rounds and it does not increase with time. This is because whether a miner wants to obtain ₿ 0.001, or ₿ 0.9, they have to find a full solution to reach this target. So the blue line really indicates the time it takes to find a full solution. Even though in expectation this takes 1000 rounds (for $\alpha_i = 0.001$), in 1% of cases a miner has to wait in excess of 4500 rounds.

It can be seen that the incentive compatible scheme requires somewhat longer to reach the same target than the proportional scheme. This is because not all reward is shared according to the reported shares, but is partly distributed over the discoverers

[11]Note that since the process of finding shares is a random process, she could never get an unconditional guarantee.

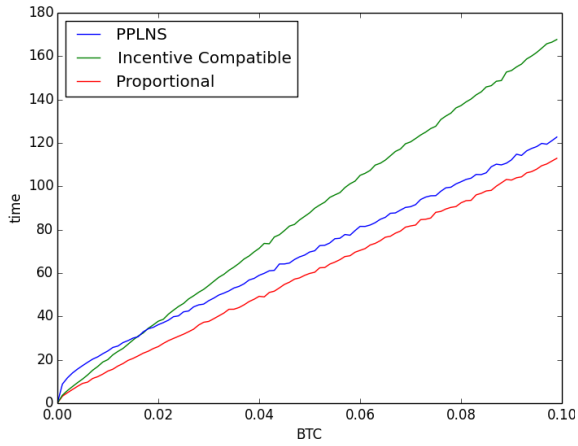



Figure 6.2: Comparison of the new incentive compatible scheme to PPLNS.

of full solutions. However, no matter what the target is, the difference in time required differs no more than a small multiplicative factor. Finally, note that since it takes longer to reach targets with high probability, the expected payouts between all three functions is the same.

In Figure 6.1b we plot the CDF of the time needed to earn ₿ 0.1. With overwhelming probability $R^{(\text{prop})}$ pays out at least ₿ 0.1 within 150 rounds, and $R^{(\text{ic})}$ pays out at least ₿ 0.1 within 200 rounds. Solo mining does not fit on this scale and it wouldn't be until around round 7,000 before a miner makes at least ₿ 0.1 with overwhelming probability.

We compare the new incentive compatible scheme to the PPLNS scheme in Figure 6.2. Here we can see that the new incentive compatible scheme performs worse by a small multiplicative factor, the PPLNS scheme performs worse by a small additive factor. This means that for small Bitcoin targets it would be faster to use the IC reward function, whereas for larger target the PPLNS reward function performs better.

From these simulations we can conclude that the trade-off for using the incentive compatible or PPLNS reward function compared to the proportional reward function is a modest delay in the time it would take miners to reach a minimal amount of bitcoin with high probability. In return we get a scheme in which it is obvious for miners what the most profitable strategy for them is.

## 6.8 Conclusions & Open Problems

We set out with a simple question: as a mining pool operator, in the absence of other mining pools or outside options, which reward functions will incentivize miners to

report full solutions immediately? In this simple model it would be reasonable to assume that miners always have an incentive to report immediately. However, we show that for proportional rewards, there are situations in which miners prefer to hold on to a full solution temporarily in order to improve their payout, harming the entire pool in the process. We also defined a novel reward function that is incentive compatible in this model (and remains so even in more powerful models). While this new scheme is not quite as efficient as proportional rewards in terms of smoothing the miners' revenue streams, it comes reasonably close in practice. We have also showed that the PPLNS reward function is incentive compatible. For a pool operator there are some tradeoffs in deciding to use our new incentive compatible scheme versus the PPLNS scheme. The latter requires a certain lead-up time, where the rewards to miners are below their fraction of the mining power. It also requires pool operators to maintain a more complex state and the payouts are arguably somewhat less transparent. On the other hand, our new incentive compatible method sometimes pays out a rather large amount to the discoverer of the full solution.

We have given a first informational model for which we can characterize incentive compatibility for all reward functions that fall in the model. We've also looked at a particular reward function that falls outside this model, and proved incentive compatibility from first principles. The next enticing question is to see if we can characterize incentive compatibility in this larger informational model at a high level, so that we can quickly identify which other reward functions would be incentive compatible. There are many reward functions in use today [Rosenfeld, 2011] that are not covered by any of our results. For example, the Geometric Method weights shares differently according to the order of shares in a round and Slush's Method takes the time of reported shares in a round into account. Defining a common informational model, characterizing incentive compatibility in this model, and classifying these methods remains an interesting open problem.

We stress that our incentive-compatible reward function will remain so even in a model with more extensive history transcripts. Our goal was to introduce the first rigorous, although simplified by omitting notions of time or order of share reporting, model of Bitcoin mining pools and demonstrate that even this simple model can lead to non-intuitive results.

# Chapter Appendix

## 6.A Omitted Proofs

### 6.A.1 Proof of Lemma 6.4

For a reward function $R$, a player $i$ has an incentive to report full solutions immediately, iff the following condition holds for all $\{\alpha_i\}_{i=1}^n, \mathbf{b}_t, D, i$:

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) \;\leq\; \frac{\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]}{D} \tag{6.4}$$

*Proof.* ($\Rightarrow$) This direction is straightforward: when it is beneficial to delay until 1 more share is reported, then there exists a profitable delay (namely $d = 1$).

($\Leftarrow$) We need to prove that for all $d$, the following inequality holds:

$$\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1 = d} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)) \;\leq\; \frac{d}{D}\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right].$$

We prove this by induction on $d$, where the induction hypothesis is equation (6.2). For the base case $d = 1$ the statement follows directly from condition (6.3). So consider the case $d > 1$:

$$
\begin{aligned}
&\sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{b}) - R_i(\mathbf{b}_t)) \\
&= \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \left( R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t) \right. \\
&\qquad \left. + \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d-1} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j + \mathbf{b}) - R_i(\mathbf{b}_t + \mathbf{e}_j)) \right) \\
&\leq \frac{1}{D}\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \\
&\qquad + \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \sum_{\mathbf{b} \text{ s.t. } ||\mathbf{b}||_1=d-1} \Pr(\text{seeing } \mathbf{b}) \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j + \mathbf{b}) - R_i(\mathbf{b}_t + \mathbf{e}_j)) \\
&\leq \frac{1}{D}\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] + \sum_{\mathbf{e}_j} \Pr(\text{seeing } \mathbf{e}_j) \frac{d-1}{D} \mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right] \\
&= \frac{d}{D}\mathbb{E}_{\mathbf{b}}\left[R_i(\mathbf{b})\right]
\end{aligned}
$$

where the first inequality follows from condition (6.3), and the second from the induction hypothesis. □

### 6.A.2 Proof of Lemma 6.6

The proportional rule $R_i^{(\text{prop})}(\mathbf{b}) = \frac{b_i}{||\mathbf{b}||_1}$ is not incentive compatible.

*Proof.* Instantiate (6.3) for the proportional rule. For the right hand side we have:

$$
\begin{aligned}
\mathbb{E}_{\mathbf{b}}\left[R_i^{(\text{prop})}(\mathbf{b}, s)\right]/D &= \frac{1}{D}\sum_{k=1}^{\infty} \Pr[\text{full solution is found at } k^{\text{th}} \text{ block}] \frac{\mathbb{E}[b_i|k]}{k} \\
&= \frac{1}{D}\sum_{k=1}^{\infty} \left(1 - \frac{1}{D}\right)^{k-1} \frac{1}{D} \frac{k \cdot \alpha_i}{k} \\
&= \frac{\alpha_i}{D}\sum_{k=1}^{\infty} \left(1 - \frac{1}{D}\right)^{k-1} \frac{1}{D} \\
&= \frac{\alpha_i}{D}
\end{aligned}
$$

Now for the left hand side. In the following let $k = ||\mathbf{b}_t||_1$:

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left( R_i^{(\text{prop})}(\mathbf{b}_t + \mathbf{e}_j) - R_i^{(\text{prop})}(\mathbf{b}_t) \right) \\
&= \alpha_i \cdot \left( \frac{b_i + 1}{k+1} \right) + (1 - \alpha_i) \cdot \left( \frac{b_i}{k+1} \right) - \frac{b_i}{k} \\
&= \frac{\alpha_i b_i + \alpha_i + b_i - \alpha_i b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i + b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i}{k+1} + b_i \left( \frac{1}{k+1} - \frac{1}{k} \right) \\
&= \frac{\alpha_i}{k+1} - \frac{b_i}{k(k+1)} \\
&= \frac{\alpha_i - \frac{b_i}{k}}{k+1}
\end{aligned}$$

Recall that for an incentive compatible scheme we need:

$$\begin{aligned}
\frac{\alpha_i - \frac{b_i}{k}}{k+1} &\leq \frac{\alpha_i}{D} \\
\alpha_i - \frac{b_i}{k} &\leq \alpha_i \frac{k+1}{D} \\
\frac{b_i}{k} &\geq \alpha_i \left( 1 - \frac{k+1}{D} \right).
\end{aligned}$$

This condition is not guaranteed to be satisfied. In particular, for every $\alpha_i > 0$ there exist positive values $b_i, k, D$ such that the condition is violated. □

## 6.B Incentive Compatibility with Preemptions

In Section 6.3 we showed that there is a simple condition that precisely characterizes when a reward function $R$ is incentive compatible, under the assumption that no other miner finds and reports a full solution during this delay. In reality a miner does have to take this possibility into account, so in this section we show exactly how the IC condition changes when we drop this assumption.

If we decide to delay reporting the full solution until one additional share is found, then with probability $1/D$ that share will actually be a full solution itself. Without loss of generality we may assume that this solution will be reported immediately

(otherwise we could simply ignore its effect). Recall that $\mathbf{b}_t$ is the number of reported shares per miner *including* the unreported full solution that miner $i$ has, and that $\mathbf{e}_j$ is the vector that has zeros everywhere except its $j^{\text{th}}$ component, where it is 1. So the expected payout for delaying for one round becomes:

$$\frac{1}{D}\sum_j \alpha_j R_i(\mathbf{b}_t - \mathbf{e}_i + \mathbf{e}_j) + \frac{D-1}{D}\sum_j \alpha_j R_i(\mathbf{b}_t + \mathbf{e}_j)$$

Thus the condition of incentive compatibility is:

$$\begin{aligned} &\frac{1}{D}\sum_j \alpha_j \left(R_i(\mathbf{b}_t - \mathbf{e}_i + \mathbf{e}_j) - R_i(\mathbf{b}_t)\right) \\ +\ &\frac{D-1}{D}\sum_j \alpha_j \left(R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)\right) \quad \leq \quad \frac{\mathbb{E}_{\mathbf{b}}[R_i(\mathbf{b})]}{D} \end{aligned}$$

For the reward functions that are monotonic increasing in each component (which by Lemma 6.5 are precisely the reward functions where miners always report all shares) this additional term is negative. Therefore, the IC condition is only easier to satisfy. This means that reward functions that are proven to be incentive compatible using Lemma 6.4 are still incentive compatible. However, one might worry that our proof that the proportional reward function is not incentive compatible might break. We show next that the threat of being scooped actually does not impact the result qualitatively.

### 6.B.1 Proportional

For the proportional reward function we can instantiate the left-hand side as (taking $||\mathbf{b}_t||_1 = k$):

$$\begin{aligned} &\frac{1}{D}\left(\alpha_i\left(\frac{b_i}{k} - \frac{b_i}{k}\right) + (1-\alpha_i)\left(\frac{b_i - 1}{k} - \frac{b_i}{k}\right)\right) \\ +\ &\frac{D-1}{D}\left(\alpha_i\left(\frac{b_i + 1}{k+1} - \frac{b_i}{k}\right) + (1-\alpha_i)\left(\frac{b_i}{k+1} - \frac{b_i}{k}\right)\right) \\ =\ &\frac{1}{D}\frac{1-\alpha_i}{k} + \frac{D-1}{D}\frac{\alpha_i - \frac{b_i}{k}}{k+1} \end{aligned}$$

So the proportional reward function is IC if and only if

$$\frac{1}{D}\frac{1-\alpha_i}{k} + \frac{D-1}{D}\frac{\alpha_i - \frac{b_i}{k}}{k+1} \leq \frac{\alpha_i}{D}.$$

Again this is not guaranteed to be satisfied.So including the possibility of another miner finding a full solution does not qualitatively change the incentive compatibility results, although quantitatively there may be situations where a miner would choose to delay if she does not fear being scooped, but choose to report if she does include this possibility.

## 6.C Multiple Pools

In the main text we've assumed that there are no other pools that compete for finding solutions to the cryptographic puzzle. This is reasonable from the perspective of proving positive results: any incentive compatible scheme should be incentive compatible regardless of how much mining power other pools have.

However, to convincingly reject the proportional rule as not incentive compatible, we should take the effect of other pools into account. In the following let $\sum_{i=1}^{n} \alpha_i = \alpha_P < 1$ be the total mining power of the pool, so all other mining power —of both other pools and solo miners— is $1 - \alpha_P$. For notational simplicity we do not consider being scooped by a different miner in our own pool; it's obvious how this can be included by comparing the results to the one in Appendix 6.B. When we consider to delay reporting a full solution until one more share is found —either inside or outside the pool— then our expected utility for doing so is

$$\sum_j \alpha_j R_i(\mathbf{b}_t + \mathbf{e}_j) + (1 - \alpha_P) \left( \frac{1}{D} \cdot 0 + \frac{D-1}{D} R_i(\mathbf{b}_t) \right).$$

We don't really care if some other pool finds another share. This does not affect us. But if another pool finds a full solution and reports it, then our mining pool misses out on a complete payment that it could have received. So the condition for incentive compatibility becomes

$$\sum_{j=1}^{n} \alpha_j \cdot (R_i(\mathbf{b}_t + \mathbf{e}_j) - R_i(\mathbf{b}_t)) - (1 - \alpha_P) \frac{R_i(\mathbf{b})}{D} \;\leq\; \alpha_P \frac{\mathbb{E}_{\mathbf{b}} \left[ R_i(\mathbf{b}_t) \right]}{D}.$$

Under the assumption that the pool in expectation will collect $\alpha_P$ of the total reward among pools, and that miner $i$ collects $\frac{\alpha_i}{\alpha_P}$ of the pool she is in, the right-hand side will remain $\frac{\alpha_i}{D}$. The new term on the left-hand side is simply $(1 - \alpha_P)\frac{b_i}{kD}$. The

other term on the left-hand side changes slightly:

$$\begin{aligned}
&\sum_{j=1}^{n} \alpha_j \cdot \left( R_i^{(\text{prop})}(\mathbf{b}_t + \mathbf{e}_j) - R_i^{(\text{prop})}(\mathbf{b}_t) \right) \\
&= \alpha_i \cdot \left( \frac{b_i + 1}{k+1} \right) + (\alpha_P - \alpha_i) \cdot \left( \frac{b_i}{k+1} \right) - \frac{b_i}{k} \\
&= \frac{\alpha_i b_i + \alpha_i + \alpha_P b_i - \alpha_i b_i}{k+1} - \frac{b_i}{k} \\
&= \frac{\alpha_i + \alpha_P b_i}{k+1} - \frac{b_i}{k}.
\end{aligned}$$

This cannot be simplified to the same convenient expression we had in Section 6.3. Combining these terms the condition for incentive compatibility of the proportional reward function becomes:

$$\frac{\alpha_i + \alpha_P b_i}{k+1} - \frac{b_i}{k} - (1 - \alpha_P)\frac{b_i}{kD} \leq \frac{\alpha_i}{D}$$

and after rewriting this:

$$\frac{\alpha_i}{k+1} + \alpha_P b_i \left( \frac{1}{kD} + \frac{1}{k+1} \right) - \frac{b_i}{k} \left( 1 + \frac{1}{D} \right) \leq \frac{\alpha_i}{D}$$

# Bibliography


Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 1–, New York, NY, USA, 2004. ACM. ISBN 1-58113-838-5. doi: 10.1145/1015330.1015430. URL `http://doi.acm.org/10.1145/1015330.1015430`.

Jacob Abernethy, Yiling Chen, and Jennifer Wortman Vaughan. Efficient market making via convex optimization, and a connection to online learning. *ACM Transactions on Economics and Computation*, 1(2):12, 2013.

S. N. Afriat. The construction of utility functions from expenditure data. *International Economic Review*, 8(1):pp. 67–77, 1967. ISSN 00206598. URL `http://www.jstor.org/stable/2525382`.

Charu C. Aggarwal. *Outlier Analysis*. Springer New York, 2013.

Ravindra K. Ahuja and James B. Orlin. Inverse optimization. *Operations Research*, 49(5):771–783, 2001. doi: 10.1287/opre.49.5.771.10607.

Shun-ichi Amari and Hiroshi Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.

E. Anshelevich, A. Dasgupta, J. Kleinberg, . Tardos, T. Wexler, and T. Roughgarden. The price of stability for network design with fair cost allocation. *SIAM Journal on Computing*, 38(4):1602–1623, 2008. doi: 10.1137/070680096. URL `http://dx.doi.org/10.1137/070680096`.

Pablo Daniel Azar, Constantinos Daskalakis, Silvio Micali, and S. Matthew Weinberg. Optimal and efficient parametric auctions. In *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 596–604, 2013.

Moshe Babaioff, Robert D. Kleinberg, and Aleksandrs Slivkins. Truthful mechanisms with implicit payment computation. In *Proceedings of the 11th ACM Conference*

*on Electronic Commerce*, EC '10, pages 43–52, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-822-3. doi: 10.1145/1807342.1807349. URL `http://doi.acm.org/10.1145/1807342.1807349`.

Moshe Babaioff, Liad Blumrosen, Shaddin Dughmi, and Yaron Singer. Posting prices with unknown distributions. In *Innovations in Computer Science (ICS)*. Tsinghua University Press, January 2011.

Patrick Bajari, Han Hong, and Stephen P Ryan. Identification and estimation of a discrete game of complete information. *Econometrica*, 78(5):1529–1568, 2010.

Maria-Florina Balcan, Avrim Blum, Jason D. Hartline, and Yishay Mansour. Reducing mechanism design to algorithm design via machine learning. *J. Comput. Syst. Sci.*, 74(8):1245–1270, 2008.

S. Baliga and R. Vohra. Market research and market design. *Advances in Theoretical Economics*, 3, 2003. Article 5.

M. J. Bayarri and M. H. DeGroot. Optimal reporting of predictions. *Journal of the American Statistical Association*, 84(405):214–222, 1989. doi: 10.1080/01621459.1989.10478758.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Commun. ACM*, 18(9):509–517, September 1975. ISSN 0001-0782.

J Eric Bickel and Seong Dae Kim. Verification of the weather channel probability of precipitation forecasts. *Monthly Weather Review*, 136(12):4867–4881, 2008.

John P Bonin. On the design of managerial incentive structures in a decentralized planning environment. *The American Economic Review*, 66(4):682–687, 1976.

Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. In *2015 IEEE Symposium on Security and Privacy*, May 2015. URL `http://www.jbonneau.com/doc/BMCNKF15-IEEESP-bitcoin.pdf`.

Craig Boutilier. Eliciting forecasts from self-interested experts: scoring rules for decision makers. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 737–744. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

Leo Breiman. Random forests. *Machine Learning*, pages 5–32, 2001.

Timothy F Bresnahan and Peter C Reiss. Empirical models of discrete games. *Journal of Econometrics*, 48(1):57–81, 1991.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Optics-of: Identifying local outliers. In *PKDD*, pages 262–270, 1999.

Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104, 2000.

Glenn W Brier. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

Michael Brückner and Tobias Scheffer. Stackelberg games for adversarial prediction problems. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 547–555. ACM, 2011.

Kevin Buchin, Olivier Devillers, Wolfgang Mulzer, Okke Schrijvers, and Jonathan Shewchuk. Vertex deletion for 3d delaunay triangulations. In *European Symposium on Algorithms*, pages 253–264. Springer Berlin Heidelberg, 2013.

Andreas Buja, Werner Stuetzle, and Yi Shen. Loss functions for binary class probability estimation and classification: Structure and applications. 2005.

Yang Cai, Constantinos Daskalakis, and Christos H Papadimitriou. Optimum statistical estimation with strategic data sources. In *COLT*, pages 280–296, 2015.

N. Cesa-Bianchi, C. Gentile, and Y. Mansour. Regret minimization for reserve prices in second-price auctions. *IEEE Transactions on Information Theory*, 61(1):549–564, 2015.

Nicolo Cesa-Bianchi, Yishay Mansour, and Gilles Stoltz. Improved second-order bounds for prediction with expert advice. *Machine Learning*, 66(2-3):321–352, 2007.

Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

Moses Charikar, Chandra Chekuri, Ashish Goel, Sudipto Guha, and Serge Plotkin. Approximating a finite metric by a small number of tree metrics. *Proceedings of Foundations of Computer Science*, pages 379–388, 1998.

Shuchi Chawla, Jason D. Hartline, and Denis Nekipelov. Mechanism design for data science. *CoRR*, abs/1404.5971, 2014. URL `http://arxiv.org/abs/1404.5971`.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player Nash equilibria. *Journal of the ACM (JACM)*, 56(3):14, 2009.

Yiling Chen and David M Pennock. Designing markets for prediction. *AI Magazine*, 31(4):42–52, 2010.

Alessandro Chiesa, Silvio Micali, and Zeyuan Allen Zhu. Mechanism design with approximate valuations. In *Innovations in Theoretical Computer Science 2012, Cambridge, MA, USA, January 8-10, 2012*, pages 34–38, 2012.

Byung-Gon Chun, Kamalika Chaudhuri, Hoeteck Wee, Marco Barreno, Christos H. Papadimitriou, and John Kubiatowicz. Selfish caching in distributed systems: A game-theoretic analysis. In *Proceedings of the 23rd Annual ACM Symposium on PODC*, PODC '04, pages 21–30, New York, NY, USA, 2004. ACM. ISBN 1-58113-802-4. doi: 10.1145/1011767.1011771. URL `http://doi.acm.org/10.1145/1011767.1011771`.

Richard Cole and Tim Roughgarden. The sample complexity of revenue maximization. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 243–252. ACM, 2014.

Nicolas T Courtois and Lear Bahack. On subversive miner strategies and block withholding attack in bitcoin digital currency. *arXiv preprint arXiv:1402.1718*, 2014.

Jacques Crémer and Richard P. McLean. Optimal selling strategies under uncertainty for a discriminating monopolist when demands are interdependent. *Econometrica*, 53(2):345–361, 1985.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. In *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*, STOC '06, pages 71–78, New York, NY, USA, 2006. ACM. ISBN 1-59593-134-1. doi: 10.1145/1132516.1132527. URL `http://doi.acm.org/10.1145/1132516.1132527`.

Tamraparni Dasu, Shankar Krishnan, Suresh Venkatasubramanian, and Ke Yi. An information-theoretic approach to detecting changes in multi-dimensional data streams. In *In Proc. Symp. on the Interface of Statistics, Computing Science, and Applications*. Citeseer, 2006.

Ofer Dekel, Felix Fischer, and Ariel D Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76(8):759–777, 2010.

Nikhil R Devanur, Jason D Hartline, and Qiqi Yan. Envy freedom and prior-free mechanism design. *Journal of Economic Theory*, 2014.

Nikhil R. Devanur, Zhiyi Huang, and Christos-Alexandros Psomas. The sample complexity of auctions with side information. 2016. To appear in *STOC '16*.

Peerapong Dhangwatnotai, Tim Roughgarden, and Qiqi Yan. Revenue maximization with a single sample. In *Proceedings of the 11th ACM Conference on Electronic Commerce (EC)*, pages 129–138, 2010.

S. Dughmi, L. Han, and N. Nisan. Sampling and representation complexity of revenue maximization. In *Workshop on Internet and Network Economics (WINE)*, pages 277–291, 2014.

Aryeh Dvoretzky, Jack Kiefer, and Jacob Wolfowitz. Asymptotic minimax character of the sample distribution function and of the classical multinomial estimator. *The Annals of Mathematical Statistics*, pages 642–669, 1956.

Benjamin Edelman, Michael Ostrovsky, and Michael Schwarz. Internet advertising and the generalized second-price auction: Selling billions of dollars worth of keywords. *American Economic Review*, 97(1):242–259, 2007. doi: 10.1257/aer.97.1.242. URL `http://www.aeaweb.org/articles.php?doi=10.1257/aer.97.1.242`.

Pavlos S. Efraimidis and Paul G. Spirakis. Weighted random sampling with a reservoir. *Information Processing Letters*, 97(5):181–185, 2006.

Edith Elkind. Designing and learning optimal finite support auctions. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 736–745. SIAM, 2007.

Andrew F. Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. Systematic construction of anomaly detection benchmarks from real data. In *ACM SIGKDD Workshop on Outlier Detection and Description*, pages 16–21, 2013.

Eleazar Eskin, Andrew Arnold, Michael Prerau, Leonid Portnoy, and Sal Stolfo. A geometric framework for unsupervised anomaly detection. In Daniel Barbará and Sushil Jajodia, editors, *Applications of Data Mining in Computer Security*, pages 77–101, Boston, MA, 2002.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, volume 96, pages 226–231, 1996.

Ittay Eyal. The Miner's Dilemma. In *IEEE Symposium on Security and Privacy*, 2015.

Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. In *Financial Cryptography*, 2014.

R. A. Finkel and J. L. Bentley. Quad trees a data structure for retrieval on composite keys. *Acta Informatica*, 4(1):1–9, 1974.

Dean P. Foster and Rakesh V. Vohra. Calibrated learning and correlated equilibrium. *Games and Economic Behavior*, 21(12):40 – 55, 1997. ISSN 0899-8256. doi: http://dx.doi.org/10.1006/game.1997.0595. URL `http://www.sciencedirect.com/science/article/pii/S0899825697905959`.

Dimitris Fotakis, Spyros Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul Spirakis. The structure and complexity of Nash equilibria for a selfish routing game. In *Automata, Languages and Programming*, volume 2380 of *Lecture Notes in Computer Science*, pages 123–134. Springer Berlin Heidelberg, 2002. ISBN 978-3-540-43864-9. doi: 10.1007/3-540-45465-9_12. URL `http://dx.doi.org/10.1007/3-540-45465-9_12`.

Peter Frazier, David Kempe, Jon Kleinberg, and Robert Kleinberg. Incentivizing exploration. In *Proceedings of the fifteenth ACM conference on Economics and computation*, pages 5–22. ACM, 2014.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Hu Fu, Nima Haghpanah, Jason D. Hartline, and Robert Kleinberg. Optimal auctions for correlated buyers with sampling. In *EC*, pages 23–36, 2014.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Andrew V Goldberg and Jason D Hartline. Collusion-resistant mechanisms for single-parameter agents. In *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 620–629. Society for Industrial and Applied Mathematics, 2005.

Andrew V Goldberg, Jason D Hartline, Anna R Karlin, Michael Saks, and Andrew Wright. Competitive auctions. *Games and Economic Behavior*, 55(2):242–269, 2006.

Yannai A Gonczarowski and Noam Nisan. Efficient empirical revenue maximization in single-parameter auction environments. In *Proceedings of the annual 49th ACM symposium on Theory of computing*, page to appear. ACM, 2017.

Sudipto Guha, Adam Meyerson, Nina Mishra, Rajeev Motwani, and Liadan O'Callaghan. Clustering data streams: Theory and practice. *IEEE Trans. Knowl. Data Eng.*, 15(3):515–528, 2003.

Sudipto Guha, Nina Mishra, Roy Gourav, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. In *Proceedings of The 33rd International Conference on Machine Learning*, pages 2712–2721, 2016.

Antonin Guttman. R-trees: A dynamic index structure for spatial searching. In *SIGMOD*, pages 47–57, 1984.

Bach Q Ha and Jason D Hartline. Mechanism design via consensus estimates, cross checking, and profit extraction. *ACM Transactions on Economics and Computation*, 1(2):8, 2013.

Moritz Hardt, Nimrod Megiddo, Christos Papadimitriou, and Mary Wootters. Strategic classification. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 111–122. ACM, 2016.

Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000. ISSN 1468-0262. doi: 10.1111/1468-0262.00153. URL `http://dx.doi.org/10.1111/1468-0262.00153`.

Jason Hartline. Mechanism design and approximation. Book draft (September 2014), September 2014.

Jason D Hartline and Tim Roughgarden. Optimal mechanism design and money burning. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 75–84. ACM, 2008.

Christian Franz Horn, Bjoern Sven Ivens, Michael Ohneberg, and Alexander Brem. Prediction markets–a literature review 2014. *The Journal of Prediction Markets*, 8 (2):89–126, 2014.

Joseph T Howson Jr. Equilibria of polymatrix games. *Management Science*, 18 (5-part-1):312–318, 1972.

Hao Huang and Shiva Prasad Kasiviswanathan. Streaming anomaly detection using randomized matrix sketching. *Proceedings of the VLDB Endowment*, 9(3):192–203, 2015.

Zhiyi Huang, Yishay Mansour, and Tim Roughgarden. Making the most of your samples. *Proceedings of EC'15*, 2015.

Kamal Jain and Mohammad Mahdian. Cost sharing. *Algorithmic game theory*, pages 385–410, 2007.

Benjamin Johnson, Aron Laszka, Jens Grossklags, Marie Vasek, and Tyler Moore. Game-theoretic analysis of DDoS attacks against Bitcoin mining pools. In *Workshop on Bitcoin Research*, 2014.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics 26. Providence, RI: American Mathematical Society*, 1984.

Victor Richmond R Jose, Robert F Nau, and Robert L Winkler. Scoring rules, generalized entropy, and utility maximization. *Operations research*, 56(5):1146–1157, 2008.

Sham M Kakade, Adam Tauman Kalai, and Katrina Ligett. Playing games with approximation algorithms. *SIAM Journal on Computing*, 39(3):1088–1106, 2009.

S. Kalyanaraman and C. Umans. The complexity of rationalizing network formation. In *Foundations of Computer Science, 2009. FOCS '09. 50th Annual IEEE Symposium on*, pages 485–494, Oct 2009. doi: 10.1109/FOCS.2009.48.

Shankar Kalyanaraman and Christopher Umans. The complexity of rationalizing matchings. In Seok-Hee Hong, Hiroshi Nagamochi, and Takuro Fukunaga, editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 171–182. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-92181-3. doi: 10.1007/978-3-540-92182-0_18. URL `http://dx.doi.org/10.1007/978-3-540-92182-0_18`.

Michael Kearns, Michael L. Littman, and Satinder Singh. Graphical models for game theory. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, UAI'01, pages 253–260, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1-55860-800-1. URL `http://dl.acm.org/citation.cfm?id=2074022.2074054`.

Daniel Kifer, Shai Ben-David, and Johannes Gehrke. Detecting change in data streams. In *VLDB*, pages 180–191, 2004.

Robert D. Kleinberg and Frank Thomson Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *FOCS*, pages 594–605, New York, New York, USA., 2003. IEEE Computer Society. URL `http://dblp.uni-trier.de/db/conf/focs/focs2003.html#KleinbergL03`.

Edwin M Knorr and Raymond T Ng. A unified notion of outliers: Properties and computation. In *KDD*, pages 219–222, 1997.

Edwin M Knorr and Raymond T Ng. Algorithms for mining distancebased outliers in large datasets. In *VLDB*, pages 392–403, 1998.

Edwin M Knorr and Raymond T Ng. Finding intensional knowledge of distance-based outliers. In *VLDB*, volume 99, pages 211–222, 1999.

Edwin M Knorr, Raymond T Ng, and Vladimir Tucakov. Distance-based outliers: algorithms and applications. *VLDB Journal*, 8(3-4):237–253, 2000.

Volodymyr Kuleshov and Okke Schrijvers. *Inverse Game Theory: Learning Utilities in Succinct Games*, pages 413–427. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015. ISBN 978-3-662-48995-6.

Aron Laszka, Benjamin Johnson, and Jens Grossklags. When bitcoin mining pools run dry. In *International Conference on Financial Cryptography and Data Security*, pages 63–77. Springer, 2015.

Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms-the numenta anomaly benchmark. *arXiv:1510.03336*, 2015.

Stephen T Lawless. Crying wolf: false alarms in a pediatric intensive care unit. *Critical care medicine*, 22(6):981–985, 1994.

Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 919–927. International Foundation for Autonomous Agents and Multiagent Systems, 2015.

T. Lindvall. *Lectures on the coupling method*. Wiley, New York, 1992.

Wietze Lise. Estimating a game theoretic model. *Computational Economics*, 18(2): 141–157, 2001.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1):3:1–3:39, March 2012.

Yang Liu and Yiling Chen. A bandit framework for strategic regression. In *Advances in Neural Information Processing Systems*, pages 1813–1821, 2016.

Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, and Aquinas Hobor. On power splitting games in distributed computation: The case of bitcoin pooled mining. Technical report, Cryptology ePrint Archive, Report 2015/155, 2015, http://eprint. iacr. org, 2015.

Mohammad Mahdian, Okke Schrijvers, and Sergei Vassilvitskii. Algorithmic cartography: Placing points of interest and ads on maps. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 755–764. ACM, 2015.

Yishay Mansour, Aleksandrs Slivkins, Vasilis Syrgkanis, and Zhiwei Steven Wu. Bayesian exploration: Incentivizing exploration in bayesian games. *arXiv preprint arXiv:1602.07570*, 2016.

Pascal Massart. The tight constant in the Dvoretzky-Kiefer-Wolfowitz inequality. *The Annals of Probability*, pages 1269–1283, 1990.

John McCarthy. Measures of the value of information. *Proceedings of the National Academy of Sciences of the United States of America*, 42(9):654, 1956.

Andres Munoz Medina and Mehryar Mohri. Learning theory and algorithms for revenue optimization in second price auctions with reserve. In *Proceedings of The 31st Intl. Conf. on Machine Learning*, pages 262–270, 2014.

Edgar C Merkle and Mark Steyvers. Choosing a strictly proper scoring rule. *Decision Analysis*, 10(4):292–304, 2013.

Andrew Miller, Elaine Shi, Ahmed Kosba, and Jonathan Katz. Nonoutsourceable Scratch-Off Puzzles to Discourage Bitcoin Mining Coalitions (preprint), 2014.

Nolan Miller, Paul Resnick, and Richard Zeckhauser. Eliciting informative feedback: The peer-prediction method. *Management Science*, 51(9):1359–1373, 2005.

J. Morgenstern and T. Roughgarden. The psuedo-dimension of near-optimal auctions. To appear in NIPS '16, 2015.

Hervé Moulin. Incremental cost sharing: Characterization by coalition strategy-proofness. *Social Choice and Welfare*, 16(2):279–320, 1999.

Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1): 58–73, 1981.

Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1 (2012):28, 2008.

John Nash. Non-cooperative games. *Annals of mathematics*, pages 286–295, 1951.

Z. Neeman. The effectiveness of English auctions. *Games and Economic Behavior*, 43(2):214–238, 2003.

Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, pages 1–18, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3410-5. doi: 10.1145/2764468.2764522. URL `http://doi.acm.org/10.1145/2764468.2764522`.

Andrew Y. Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proc. 17th International Conf. on Machine Learning*, pages 663–670. Morgan Kaufmann, 2000.

Noam Nisan. Introduction to mechanism design (for computer scientists). In Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.

Christos H. Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *J. ACM*, 55(3):14:1–14:29, August 2008. ISSN 0004-5411. doi: 10.1145/1379759.1379762. URL `http://doi.acm.org/10.1145/1379759.1379762`.

Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv:1112.4980*, 2011.

RobertW. Rosenthal. A class of games possessing pure-strategy Nash equilibria. *International Journal of Game Theory*, 2(1):65–67, 1973. ISSN 0020-7276. doi: 10.1007/BF01737559. URL `http://dx.doi.org/10.1007/BF01737559`.

Tim Roughgarden. *Twenty Lectures on Algorithmic Game Theory*. Cambridge University Press, 2016.

Tim Roughgarden and Okke Schrijvers. Network cost-sharing without anonymity. *ACM Transactions on Economics and Computation*, 4(2):8, 2016a.

Tim Roughgarden and Okke Schrijvers. Ironing in the dark. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, EC '16, pages 1–18, New York, NY, USA, 2016b. ACM. ISBN 978-1-4503-3936-0. doi: 10.1145/2940716.2940723. URL `http://doi.acm.org/10.1145/2940716.2940723`.

Tim Roughgarden and Okke Schrijvers. Online prediction with selfish experts. *arXiv preprint arXiv:1702.03615*, 2017.

Tim Roughgarden and Eva Tardos. Introduction to the inefficiency of equilibria. *Algorithmic Game Theory*, 17:443–459, 2007.

Paul A. Samuelson. Consumption theory in terms of revealed preference. *Economica*, 15(60):pp. 243–253, 1948. ISSN 00130427. URL `http://www.jstor.org/stable/2549561`.

Leonard J Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.

Mark J Schervish. A general method for comparing probability assessors. *The Annals of Statistics*, pages 1856–1879, 1989.

Okke Schrijvers and Jarke J van Wijk. Visual explanation of the complexity in julia sets. In *Computer Graphics Forum*, volume 32, pages 431–440. Blackwell Publishing Ltd, 2013.

Okke Schrijvers, Frits van Bommel, and Kevin Buchin. Delaunay triangulations on the word ram: Towards a practical worst-case optimal algorithm. In *Voronoi Diagrams in Science and Engineering (ISVD), 2013 10th International Symposium on*, pages 7–15. IEEE, 2013.

Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In *Financial Cryptography and Data Security*, 2016.

I. Segal. Optimal pricing mechanisms with unknown demand. *American Economic Review*, 93(3):509–529, 2003.

Nihar Bhadresh Shah and Denny Zhou. Double or nothing: Multiplicative incentive mechanisms for crowdsourcing. In *Advances in neural information processing systems*, pages 1–9, 2015.

Balasubramanian Sivan and Vasilis Syrgkanis. Vickrey auctions for irregular distributions. In *Web and Internet Economics*, pages 422–435. Springer, 2013.

Swee Chuan Tan, Kai Ming Ting, and Fei Tony Liu. Fast anomaly detection for streaming data. In *IJCAI*, pages 1511–1516, 2011.

William Thomson. Eliciting production possibilities from a well-informed manager. *Journal of Economic Theory*, 20(3):360–380, 1979.

Christine L Tsien and James C Fackler. Poor prognosis for existing monitors in the intensive care unit. *Critical care medicine*, 25(4):614–619, 1997.

Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11): 1134–1142, 1984.

Hal R. Varian. The nonparametric approach to demand analysis. *Econometrica*, 50(4):pp. 945–973, 1982. ISSN 00129682. URL `http://www.jstor.org/stable/1912771`.

Hal R Varian. Revealed preference. *Samuelsonian Economics and the Twenty-First Century*, 2006.

Hal R Varian. Position auctions. *international Journal of industrial Organization*, 25 (6):1163–1178, 2007.

Jeffrey S. Vitter. Random sampling with a reservoir. *ACM Transactions on Mathematical Software*, 11(1):3757, 1985.

John von Neumann and Oskar Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

Kevin Waugh, Brian D. Ziebart, and J. Andrew Bagnell. Computational rationalization: The inverse equilibrium problem. 2013.

Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.

Ji Zhang and Hai Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, 10 (3):333–355, 2006.

Brian D. Ziebart, Andrew Maas, J. Andrew Bagnell, and Anind K. Dey. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *Proc. Ubicomp*, pages 322–331, 2008.