# Designing and Analyzing Machine Learning Algorithms in the Presence of Strategic Behavior

**Hanrui Zhang**

CMU-CS-23-116

July 2023

Computer Science Department
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

**Thesis Committee:**
Vincent Conitzer, Chair
Maria-Florina Balcan
Tuomas Sandholm
Nika Haghtalab (UC Berkeley)
Vahab Mirrokni (Google Research)
Renato Paes Leme (Google Research)

*Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.*



This research was sponsored by: Duke University under award number 313000039; the Center for Emerging Risk Research; the Cooperative AI Foundation; the National Science Foundation under award numbers IIS-1814056 and IIS-1527434; and the Army Research Office under award number W911NF2110230. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of any sponsoring institution, the U.S. government or any other entity.

**Keywords:** Machine Learning, Mechanism Design, Sample Complexity, Markov Decision Processes

*To my family*

# Abstract

Machine learning algorithms now play a major role in all kinds of decision-making scenarios, such as college admissions, credit approval, and resume screening. When the stakes are high, self-interested agents — about whom decisions are being made — are increasingly tempted to manipulate the machine learning algorithm, in order to better fulfill their own goals, which are generally different from the decision maker's. The fact that many machine learning algorithms can be manipulated also raises a fairness concern, since algorithms that are manipulable tend to be manipulated most effectively by those who have more resources and who are already entrenched in the system. All this highlights the importance of making machine learning algorithms *robust against manipulation*. The main focus of this dissertation is on **designing and analyzing machine learning algorithms that are robust against *strategic* manipulation**, which is different from the relatively well-studied notion of adversarial robustness.

This dissertation sets the foundations for several key problems in machine learning in the presence of strategic behavior:

- *Empirical risk minimization and generalization in classification problems* [66, 111, 115]: Traditional wisdom suggests that a classifier trained on historical observations (i.e., an *empirical risk minimizer*) usually also works well on future data points to be classified. Is this still true in the presence of strategic manipulation?
- *Distinguishing distributions with samples* [113, 114, 116]: Due to various constraints, often we have to judge the "quality" of an agent based on a few samples (e.g., screening job candidates based on a few selected papers). How should we calibrate our judgment when these samples are strategically selected or transformed?
- *Planning in Markov decision processes* [110, 117, 118]: Dynamic decision-making problems (traditionally modeled using *Markov decision processes*) can be solved efficiently when the decision maker always has complete and reliable information about the state of the world, as well as full control over which actions to take. What happens when the state of the world is reported by a strategic agent, or when a self-interested agent may interfere with the actions taken?

# Acknowledgments

First and foremost, I would like to thank my advisor, Vincent Conitzer. Vince has been a terrific advisor. Over the past six years, he has taught me everything I ever needed to conduct research — including how to ask the right questions, how to turn vague ideas into concrete progress, how to effectively communicate and collaborate with my fellow researchers, and how to manage exhaustion and frustration — often before I realized how important those things really were. Vince's support also kept me from the (arguably) less exciting aspects of research, such as funding and administrative issues.

I also want to thank the other members of my thesis committee: Nina Balcan, Nika Haghtalab, Vahab Mirrokni, Renato Paes Leme, and Tuomas Sandholm. They have provided valuable feedback on this dissertation, and offered priceless advice more generally on research and career planning.

Special thanks go to Wei Chen, Ran Duan, Jian Li, Ariel Procaccia and Pingzhong Tang, who nurtured my interest in research when I was in college. I would not be who I am without their guidance and advice.

I would like to thank my other collaborators and mentors: Lijie Chen, Yu Cheng, Yuan Deng, Simon S. Du, Fei Fang, Gagan Goel, Steven Jecmen, Zhipeng Jia, Anilesh Krishnaswamy, Haoming Li, Yi Li, Ryan Liu, Yuping Luo, Mohammad Mahdian, Jieming Mao, Benjamin Miller, Kamesh Munagala, Debmalya Panigrahi, David Rein, Jon Schneider, Nihar B. Shah, Shang-Hua Teng, David Thompson, David Wajc, Lirong Xia, Haifeng Xu, Yixuan Xu, Lin F. Yang, Tianyi Zhang, and Song Zuo, among others. I am grateful for the chance to work with and learn from each and everyone of you.

I would like to thank my fellow students and friends, with whom I had great fun during my six years in graduate school: Jiyao, Kangning, Ming, Ruosong, Yuan, Zhengjie, and many others.

Finally, I want to thank my family, for their continuing and unconditional love and support. To them I dedicate this dissertation.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Overview

Machine learning algorithms now play a major role in all kinds of decision-making scenarios, such as college admissions, credit approval, and resume screening. When the stakes are high, self-interested agents — about whom decisions are being made — are increasingly tempted to manipulate the machine learning algorithm, in order to better fulfill their own goals, which are generally different from the decision maker's. The fact that many machine learning algorithms can be manipulated also raises a fairness concern, since algorithms that are manipulable tend to be manipulated most effectively by those who have more resources and who are already entrenched in the system. All this highlights the importance of making machine learning algorithms *robust against manipulation*. The main focus of this dissertation is on **designing and analyzing machine learning algorithms that are robust against *strategic* manipulation**, which is different from the relatively well-studied notion of adversarial robustness (see, e.g., [22, 50, 58, 120]).

This dissertation sets the foundations for several key problems in machine learning in the presence of strategic behavior:

- *Empirical risk minimization and generalization in classification problems* [66, 111, 115]: Traditional wisdom suggests that a classifier trained on historical observations (i.e., an *empirical risk minimizer*) usually also works well on future data points to be classified. Is this still true in the presence of strategic manipulation?
- *Distinguishing distributions with samples* [113, 114, 116]: Due to various constraints, often we have to judge the "quality" of an agent based on a few samples (e.g., screening job candidates based on a few selected papers). How should we calibrate our judgment when these samples are strategically selected or transformed?
- *Planning in Markov decision processes* [110, 117, 118]: Dynamic decision-making problems (traditionally modeled using *Markov decision processes*) can be solved efficiently when the decision maker always has complete and reliable information about the state of the world, as well as full control over which actions to take. What happens when the state of the world is reported by a strategic agent, or when a self-interested agent may interfere with the actions taken?

While significant progress has been made towards making machine learning algorithms robust, the predominant notion of robustness considered in the literature is *adversarial robustness*. The

idea is to consider the worst-case situation, where every agent's only goal is to prevent the decision maker from making the right decisions, where "right" means "best serving the decision maker's purposes". One shortcoming of this notion is it is *often too pessimistic*: In reality, not *all* agents' goals are *always the exact opposite* of the decision maker's. For example, if an agent's most desired decision happens to be the right decision, then that agent actually has incentives to *help* the machine learning algorithm make the right decision. An alternative notion of robustness that more accurately captures such behavior is *robustness against strategic manipulation*, which requires that the machine learning algorithm should work well when each agent reacts rationally to the algorithm in a way that *best serves that agent's own goal*. This is the main focus of this dissertation.

## 1.1 Foundations of Strategic Classification

Classification is a central topic in machine learning whose importance cannot be overstated. Strategic manipulation of classification algorithms has drawn attention since more than a decade ago (see, e.g., [73]). However, most prior research on classification in the presence of stategic behavior had focused on various special cases of the problem, e.g., linear classifiers and/or separable cost functions. In particular, a unifying framework agnostic to specific models or assumptions had been missing when strategic manipulation is involved. Perhaps the most significant example of such a framework is the principle of *empirical risk minimization (ERM)*: The ERM principle states that in general, the decision-maker should simply find the classifier that works best on training data (e.g., historical observations), and it is expected that the good performance of such a classifier also generalizes to test data (e.g., future data points to be classified). Backed by rigorous theoretical analysis, the ERM principle underlies most successful machine learning algorithms currently in use, and is almost taken for granted by practitioners. However, in the presence of strategic manipulation, it was unclear whether such a fundamental principle is still well founded.

In a series of papers, I establish and empirically validate the ERM principle in the presence of strategic manipulation. In particular, I prove the *first generic generalization bound* for strategic classification [111], which can be viewed as a strategic counterpart of the celebrated Vapnik–Chervonenkis theory that characterizes generalization in traditional settings without strategic behavior. This very recent result has already been generalized and adapted into numerous related settings [47, 67, 70, 71, 78, 98]. Moreover, I give an *efficient algorithm for finding an empirical risk minimizer* among classifiers that are *immune to strategic manipulation*, even in general, unstructured environments [115]. I show that classifiers found in such a way are in a sense *automatically regularized*, and generalize well as long as the possible ways of strategic manipulation are rich enough [111]. This, together with complete robustness against strategic manipulation, makes such classifiers particularly desirable. Together with my collaborators, I empirically confirm the above theoretical findings in a classification setting using credit approval data where features can be strategically withheld [66].

## 1.2 Distinguishing Strategic Agents with Few Samples

One of the most basic problems in machine learning and statistics is *distinguishing distributions with samples*. Consider the following concrete setting: A company wants to evaluate an intern for potential conversion into full-time employment. Within the duration of the internship, the company observes the intern's performance in a few (say 3) tasks, and must decide whether to make a full-time offer accordingly. The chance that an *outstanding* candidate performs well in a task is higher than that of an *average* candidate performing well. So, for example, in order to hire as many outstanding candidates and as few average candidates as possible, one reasonable policy is to make an offer to the intern if and only if they perform well in at least 2 tasks.

From a more abstract perspective, the performance of the intern in each task is a *sample* of their true capability, where the latter can also be viewed as a *probability distribution*; the company's goal is to distinguish interns with outstanding capability from those with average capability, by observing samples from the distribution under evaluation. This is an extremely basic and well-studied problem in machine learning and statistics, with applications going well beyond the problem itself, since many more sophisticated algorithms in complex environments also solve the problem of distinguishing distributions from samples as a subroutine. Despite the importance of the problem, it had been barely studied in the presence of strategic manipulation. My work is among the first to provide fundamental algorithms and characterizations for distinguishing distributions with samples, when strategic agents may *select or transform samples* before submitting them to the algorithm [113, 114], or when strategic agents may *decide whether to participate* in the algorithm based on their own capability [116].

## 1.3 Dynamic Decision-Making with Strategic Agents

Markov decision processes (MDPs) are the canonical tool for modeling dynamic decision-making problems. The idea is to describe all relevant information at any time using a *state*, and consider all available *actions* in each state. Each action results in a *reward*, and changes the state in a possibly random way. To make optimal dynamic decisions, one only needs to solve the *planning* problem in the corresponding MDP — which is to find a *policy* which maximizes the total reward by choosing actions in an optimal way. Intuitively, such a policy should balance between immediate reward and long-term effects, which makes the planning problem nontrivial.

While numerous successful algorithms have been proposed for planning in variants of MDPs in the past decades, the strategic aspect had been largely overlooked. In many real-world decision-making scenarios, there are multiple self-interested parties involved, and the actions taken may affect these parties in different ways. In particular, the decision maker (i.e., the principal) may have different interests from the other participants' (i.e., the agent's). If participating in a policy hurts the agent's utility (for example, compared to the best outside option), then the agent may rationally choose to stop participating, leaving the principal in a difficult situation. Therefore, the principal needs to *prevent the agent from quitting* while maximizing the principal's own utility. Moreover, the principal may need to rely on the agent to report their private information in order to take the optimal actions, and the principal needs to *encourage the agent to report truthfully*. Such

strategic considerations introduce new challenges to the classical problem of planning in MDPs. One real-world example is a ride-sharing platform assigning tasks to drivers: It might benefit the platform in the long run to assign a driver an undesirable task, but the driver may respond to that by turning off the app and going home, which is a situation that the platform would like to avoid.

My work addresses these challenges posed by the presence of strategic behavior, by introducing novel methodologies and techniques. I show that even when the agent does not have any private information, the planning problem with participation constraints is already significantly different from, and in fact, harder than planning in classical MDPs — for example, in classical MDPs, optimal policies are without loss of generality deterministic and history-independent, while with participation constraints, optimal policies may necessarily be randomized and history-dependent [118]. Nonetheless, I give an efficient algorithm for planning with participation constraints [117], which is fundamentally different from existing planning algorithms for the classical problem. With private information, the problem becomes even harder: I show that the planning problem is *hard to approximate* when the time horizon is long, and at the same time, give an efficient and practical algorithm when the time horizon is constant [110].

## 1.4 Organization

The first part of the dissertation, consisting of Chapters 2, 3 and 4, focuses on foundations of strategic classification. Chapter 2, based on our AAAI 2021 paper [111], establishes a powerful and generic theory of generalization for strategic classification, and discusses the generalization behavior of incentive-compatible classifiers. Chapter 3, based on our AAAI 2021 paper [115], gives an efficient algorithm for the empirical risk minimization problem with incentive-compatible classifiers, i.e., the problem of computing the optimal incentive-compatible classifier on a finite set of training examples, or in Bayesian settings. Chapter 4, based on our AAAI 2021 paper [66], presents an empirical study of strategic classification on credit approval datasets, which shows how the theoretical results in Chapters 2 and 3 can be applied in practice.

The second part of the dissertation, consisting of Chapters 5, 6 and 7, focuses on sample-based classification of strategic agents. Chapter 5, based on our ICML 2019 paper [114], presents characterizations of optimal sample-based classification policies when strategic agents can sub-select the samples used for classification from a private pool of samples in response to the decision maker's policy. Chapter 6, based on our NeurIPS 2019 paper [113], presents structural and computational results in a similar setting, where strategic agents can transform private samples into signals before presenting them to the decision maker. Chapter 7, based on our AAAI 2021 paper [116], studies a setting where samples are costly to obtain, and shows how this can help the decision maker classify strategic agents with very few samples.

The third part of the dissertation, consisting of Chapters 8 and 9, focuses on dynamic decision-making with strategic agents. Chapter 8, based on our EC 2022 paper [117], studies a principal-agent planning problem where the principal (i.e., the decision maker) must incentivize the agent to keep participating, and gives an efficient algorithm for such dynamic decision-making problems. Chapter 9, based on our NeurIPS 2021 paper [110], studies a more challenging setting where the agent also has private information, and presents various computability and hardness results.

Finally, in Chapter 10, we summarize the main contributions of the dissertation and discuss several important future directions.

More broadly, I have also made contributions in:

- Other sub-areas of machine learning, including *learning valuations and preferences* [104, 107, 109, 112], and *learning influence propagation* [27, 28].
- Other sub-areas of mechanism design, including *ad auctions and autobidding* [32, 34, 35, 48], *combinatorial auctions and matching* [19, 33, 105, 106, 108], and *scientific peer review* [60, 61].

To stay focused, these results are not covered in this dissertation.

# Chapter 2

# Incentive-Aware PAC Learning

## 2.1 Introduction

Consider the following scenario. A financial institution plans to offer a product to a selected group of customers, and decides that the only thing that should matter in the selection process is the amount of money the customer currently owns in savings (possibly held at a different financial institution). Each customer may prove her savings balance by submitting a bank statement, and the number shown therein will be used for the selection. However, customers may choose to underreport their balance, by, for example, temporarily transferring their money to another account before the statement date. (We assume for the purpose of this example that overreporting is not possible.) The question is, how does this ability of customers to strategically underreport their balance affect the selection process?

The answer, as one would expect, depends on the specific criteria of the selection, which are presumably determined by the nature of the product. For instance, the product could be a secured loan that requires the customer to use the savings for collateral. In that case, customers will benefit from their balance being as high as possible, which means they would submit a statement with the full balance. As a result, the fact that customers can underreport would not affect the selection at all. Alternatively, the financial institution could be a non-profit entity that aims to make a product available only to those with *low* savings. In that case, every customer who wants access to this product would underreport her balance in order to be approved, and this would make the selection effectively meaningless.

More generally, often the right criteria for the selection may not be clear a priori, but have to be learned from observations. In such cases, the decision maker (e.g., the institution) has access to labeled historical data (e.g., the amount of savings owned by some customer, and whether the product was successful for the same customer). A classifier (e.g., selection criteria) is derived from the data and implemented, to which future data points (e.g., new customers) to be classified respond in a strategic way. The goal, naturally, is to classify future data points as accurately as possible, taking into account that their features may be strategically modified. The key challenge is for the classifier derived to *generalize* from past unmodified observations to future strategic data points. Such problems are partially captured by the PAC learning model (discussed in detail below), but

also exhibit additional complexity from *strategic manipulation* of the features, *where the nature of this manipulation is affected by the choice of classifier*. In this chapter, we investigate novel phenomena in PAC learning introduced by the presence of strategic manipulation.

**Our results.** Two central questions in PAC learning are the following: statistically, how many labeled data points does one need to observe in order to derive a classifier of a desired quality, and computationally, given this many labeled data points, how can one compute such a classifier? We answer these questions for arbitrary feature spaces and structures of strategic manipulation, by presenting an adapted version of the *empirical risk minimization (ERM)* principle, which roughly says that one should simply pick a classifier that has the best quality on the *unmodified* labeled data points that are observed. We show that our incentive-aware version of the ERM principle requires asymptotically the minimum possible number of labeled data points for any specific problem setup. This can be viewed as a strategic version of the VC theory, one of the central results in traditional PAC learning.

We further consider *incentive-compatible* classifiers, which provably prevent any kind of strategic manipulation by making revealing one's true features always optimal for any data point. Here, our most remarkable result is a *hypothesis-class-independent* bound on the number of labeled data points required to compute an incentive-compatible classifier of a desired quality. In traditional PAC learning, it is well known that without any prior knowledge about the ground truth (typically modeled by a *hypothesis class* consisting of possible classifiers to be considered), it is impossible to learn anything nontrivial, unless the number of labeled data points observed is trivially large or even infinite. By contrast, we show that in the presence of strategic manipulation, it is possible to learn a nontrivial classifier via our strategic version of the ERM principle without any such prior knowledge, except that the classifier learned has to be incentive-compatible. In other words, incentive-compatibility acts as a means of *regularization*, which provides nontrivial learning guarantees in and of itself.

Moreover, when the structure of strategic manipulation is transitive (i.e., if A can pretend to be B and B can pretend to be C, then A can pretend to be C), considering incentive-compatible classifiers is without loss of generality — this is commonly known as the *revelation principle* in economic theory. This, together with our results for incentive-compatible classifiers, implies that ERM-type learning in the presence of transitive strategic manipulation is *automatically regularized*. That is, the hypothesis-class-independent bound discussed above applies even without any exogenous requirement of incentive compatibility (since requiring this condition is without loss of generality), or any other prior knowledge.

## 2.2 Preliminaries

In this section, we review relevant definitions and results in PAC learning, and formulate the notion of strategic manipulation in classification.

### 2.2.1 Background on PAC Learning

Our problems of interest fit well with the *probably approximately correct (PAC)* learning model [100]. In PAC learning, there is a feature space $\mathcal{X}$, a label space $\mathcal{Y}$, and a joint population distribution $\mathcal{D} \in \Delta(\mathcal{X} \times \mathcal{Y})$ over the feature space and the label space. For a classifier $f : \mathcal{X} \to \mathcal{Y}$, the population loss $\ell_{\mathcal{D}}(f)$ is defined as the probability that $f$ assigns a wrong label to a random point with respect to $\mathcal{D}$, i.e.,

$$\ell_{\mathcal{D}}(f) = \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq y].$$

The goal of PAC learning, with target relative loss $\varepsilon > 0$ and failure probability $\delta > 0$, is to find a classifier $f \in \mathcal{H}$ from a predetermined hypothesis class $\mathcal{H} \subseteq \mathcal{Y}^{\mathcal{X}}$, by observing $m = m(\varepsilon, \delta)$ iid samples from $\mathcal{D}$, such that with probability at least $1 - \delta$, the population loss of $f$, $\ell_{\mathcal{D}}(f)$, is at most $\ell_{\mathcal{D}}(\mathcal{H}) + \varepsilon$, where $\ell_{\mathcal{D}}(\mathcal{H})$ is the population loss of the best classifier in $\mathcal{H}$, i.e.,

$$\ell_{\mathcal{D}}(\mathcal{H}) = \min_{f' \in \mathcal{H}} \ell_{\mathcal{D}}(f').$$

$\ell_{\mathcal{D}}(\mathcal{H})$ is sometimes called the *approximation error*, which models how well the hypothesis class $\mathcal{H}$ approximates the ground truth classifier. Throughout this chapter, we focus on binary labels, i.e., $\mathcal{Y} = \{0, 1\}$. With binary labels, a classifier $f : \mathcal{X} \to \{0, 1\}$ corresponds bijectively to a subset of $\mathcal{X}$ to which $f$ assigns label 1, i.e.,

$$\{x \in \mathcal{X} \mid f(x) = 1\}.$$

It is sometimes more convenient to deal with the subset of $\mathcal{X}$ associated with $f$. We will treat a classifier $f$ and the subset associated with $f$ interchangeably in the rest of the chapter.

The two central questions of PAC learning discussed earlier are partially answered by the ERM principle, formally defined below. Given $m$ iid samples $S = \{(x_i, y_i)\}_i \sim \mathcal{D}^m$,[1] the ERM principle finds a classifier $f \in \mathcal{H}$ which minimizes the empirical loss $\ell_S(f)$ on $S$, defined as

$$\ell_S(f) = \frac{1}{|S|} \sum_{i \in [|S|]} |f(x_i) - y_i|.$$

That is, the ERM principle finds

$$f \in \operatorname{argmin}_{f' \in \mathcal{H}} \ell_S(f').$$

This gives a concrete, though not always efficient, way of computing a classifier — it is known that with sufficiently many samples, the classifier found by the ERM principle achieves the desired loss and failure probability. Moreover, the number of samples (also known as the *sample complexity* — see below for a formal definition) required by the ERM principle is optimal up to a constant factor.[2] This sample complexity is asymptotically determined by the so-called *VC dimension* of the hypothesis class $\mathcal{H}$, defined below.

[1] We treat $S$ as an unordered set, since the order does not carry information.

[2] This is true only in the agnostic setting where $\ell_{\mathcal{D}}(\mathcal{H}) > 0$, on which we focus our attention throughout the chapter. Similar results can be established for the realizable setting where $\ell_{\mathcal{D}}(\mathcal{H}) = 0$.

**Definition 2.1** (VC Dimension)**.** A subset $S$ of the feature space $\mathcal{X}$ is *shattered* by a hypothesis class $\mathcal{H}$, if for any $T \subseteq S$, there is a classifier $f \in \mathcal{H}$ such that $f \cap S = T$. The *VC dimension* of $\mathcal{H}$, $d_{\mathrm{VC}}(\mathcal{H})$, is the cardinality of the largest subset of $\mathcal{X}$ that is shattered by $\mathcal{H}$, i.e.,

$$d_{\mathrm{VC}}(\mathcal{H}) = \max\left\{|S| \,\middle|\, |\{f \cap S \mid f \in \mathcal{H}\}| = 2^{|S|}\right\}.$$

The VC dimension captures the *capacity* of the hypothesis class $\mathcal{H}$. The larger the capacity is, the more samples it takes to find an approximately optimal classifier in $\mathcal{H}$. More precisely, the sample complexity of ERM is given by the following theorem (see, e.g., [93]).

**Theorem 2.1** (Sample Complexity of ERM)**.** *Fix a feature space $\mathcal{X}$, a population distribution $\mathcal{D}$, and a hypothesis class $\mathcal{H}$. For any $\varepsilon > 0$, $\delta > 0$, given*

$$m = O\left((d_{\mathrm{VC}}(\mathcal{H}) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples $S \sim \mathcal{D}^m$, with probability at least $1-\delta$, any classifier $f \in \mathcal{H}$ minimizing the empirical loss on $S$, i.e., $f \in \operatorname{argmin}_{f' \in \mathcal{H}} \ell_S(f')$, has population loss at most $\ell_{\mathcal{D}}(f) \leq \ell_{\mathcal{D}}(\mathcal{H}) + \varepsilon$. Moreover, finding any classifier achieving the same relative loss $\varepsilon$ and failure probability $\delta$ requires $\Omega\left((d_{\mathrm{VC}}(\mathcal{H}) + \log(1/\delta))/\varepsilon^2\right)$ iid samples.*

The above theorem says that the ERM principle finds an approximately optimal (up to an additive loss $\varepsilon$) classifier within $\mathcal{H}$, with an asymptotically optimal number of samples. In other words, the low empirical loss of the classifier found by the ERM principle generalizes with respect to the population distribution $\mathcal{D}$.

### 2.2.2 Strategic Manipulation in Classification

In the classical PAC learning model, the classifier always observes the real feature of the point being classified. However, as argued above, this is often not the case in real-life scenarios, where the point being classified may strategically modify its feature in order to receive a more desirable outcome. In this chapter, we model the data point's ability to modify its feature using a binary relation $\to$, which captures the *reporting structure* over the feature space. For $x_1, x_2 \in \mathcal{X}$, we say $x_1 \to x_2$ if a point with feature $x_1$ can pretend to have (i.e., *report*) feature $x_2$. For any $x \in \mathcal{X}$, we always have $x \to x$, corresponding to the fact that the point may choose not to modify its feature.[3] Throughout the chapter, we assume that all points being classified prefer label $1$ (corresponding to, e.g., acceptance) to $0$. As a result, fixing a misreporting structure $\to$ and a classifier $f : \mathcal{X} \to \{0, 1\}$, a point with feature $x$ will pretend to have feature $x'$ such that $f(x') = 1$ whenever possible, i.e., it will report $x' \in \operatorname{argmax}_{x'' : x \to x''} f(x'')$. Note that it is possible that $x' = x$. This can happen when $f(x) = 1$, or $f(x'') = 0$ for all $x''$ such that $x \to x''$, including $x$ itself.[4]

In the rest of the chapter, we focus on the following variant of the PAC learning model. The learning algorithm has access to $m$ iid *unmodified* samples $\{(x_i, y_i)\}_i \sim \mathcal{D}^m$, based on which a

[3] Note that this is *not* a technical requirement — all results in this chapter still hold without this assumption.

[4] A seemingly more general model is one in which there is a cost $c(x, x') \geq 0$ for a point with feature $x$ to report feature $x'$. We remark that with binary labels, it is without loss of generality to ignore this cost, since a way of reporting is feasible iff the gain of receiving label $1$ is larger than the cost of reporting.

classifier $f \in \mathcal{H}$ is computed.[5] The classifier $f$ is then deployed, and random points drawn from $\mathcal{D}$ modify their features strategically in response to the classifier. The strategic population loss of $f$, taking into consideration strategic manipulation, can be computed as

$$\widehat{\ell}_{\mathcal{D}}(f) = \Pr_{(x,y)\sim\mathcal{D}}\left[\max_{x':x\to x'} f(x') \neq y\right].$$

Again, the goal is to find a classifier $f \in \mathcal{H}$ with strategic population loss at most $\widehat{\ell}_{\mathcal{D}}(\mathcal{H}) + \varepsilon$, with probability at least $1-\delta$, where

$$\widehat{\ell}_{\mathcal{D}}(\mathcal{H}) = \min_{f'\in\mathcal{H}} \widehat{\ell}_{\mathcal{D}}(f').$$

## 2.3 Incentive-Aware Empirical Risk Minimization

In this section, we investigate the ERM principle in the presence of strategic manipulation. We first give an example, showing that the vanilla ERM principle, which ignores the strategic aspect of the problem, has poor performance in general. We then consider an incentive-aware variant of ERM, and analyze its generalization behavior.

### 2.3.1 How Vanilla ERM Fails

Consider the following example. The feature space $\mathcal{X} = \{x_1, x_2, x_3\}$, the hypothesis class $\mathcal{H} = 2^{\mathcal{X}}$, and the population distribution $\mathcal{D}$ assigns probability $0.5$ to feature-label pair $(x_1, 0)$, and probability $0.5$ to $(x_2, 1)$. The reporting structure allows (1) $x \to x$ for any $x \in \mathcal{X}$, and (2) $x_1 \to x_2$ and $x_2 \to x_3$. Note that since the feature space is extremely simple, even the power set $\mathcal{H} = 2^{\mathcal{X}}$ (which has VC dimension $d_{\mathrm{VC}}(\mathcal{H}) = 3$) exhibits decent generalization behavior *without* strategic manipulation. Recall that the vanilla ERM principle finds an arbitrary classifier which minimizes the empirical loss based on the unmodifed sample set. Suppose the number of samples $m = \omega(1)$. Then with high probability, the sample set consists of copies of $(x_1, 0)$ and $(x_2, 1)$ and nothing else. Any classifier $f$ satisfying $f(x_1) = 0$ and $f(x_2) = 1$ achieves $0$ empirical loss. However, with strategic manipulation, such an $f$ ends up assigning label $1$ to all (new) points whose (true) feature is $x_1$, since those points can report $x_2$ to fool the classifier. As a result, the strategic population loss is

$$\begin{aligned}
\widehat{\ell}_{\mathcal{D}}(f) &= \Pr_{(x,y)\sim\mathcal{D}}\left[\max_{x':x\to x'} f(x') \neq y\right] \\
&\geq 0.5 \cdot \left|\max_{x':x_1\to x'} f(x') - 0\right| = 0.5 \cdot |f(x_2) - 0| \\
&= 0.5.
\end{aligned}$$

[5]We consider scenarios where observed data points have no interest in manipulating the learning process, and thus will not modify their features — this is a standard assumption, and appears also in, e.g., [31]. It is definitely true that in many other scenarios, the samples observed may be strategically modified too. However, we do not consider this in the current chapter.

In other words, with high probability, any classifier found using the vanilla ERM principle is no better than random guessing. But in fact, the classifier $f'$ with $f'(x_1) = f'(x_2) = 0$ and $f'(x_3) = 1$ would have 0 loss with strategic behavior, because $\max_{x':x_1 \to x'} f'(x') = 0$ and $\max_{x':x_2 \to x'} f'(x') = f'(x_3) = 1$. Hence it is possible to do much better than vanilla ERM.

The above example shows that with strategic manipulation, the vanilla ERM principle fails spectacularly even in extremely simple cases that would be trivial without strategic manipulation. This observation aligns with the intuition that any learning procedure achieving nontrivial performance must exploit the reporting structure.

### 2.3.2 The Incentive-Aware ERM Principle

Below we present an incentive-aware version of the ERM principle (henceforth IA ERM), and show that (1) IA ERM finds a classifier with the desired properties, and (2) the sample complexity of IA ERM is asymptotically optimal. These properties of IA ERM closely resemble those of its classical counterpart without strategic manipulation, and suggests that IA ERM is the right version of the ERM principle in the strategic setting that we consider.

**Incentive-aware ERM.** The idea is to minimize the *strategic* empirical loss $\widehat{\ell}_S(f)$ on the sample set $S = \{(x_i, y_i)\}_i$, computed by replacing the true feature of every sample point with the most beneficial feature that the point can report, i.e.,

$$\widehat{\ell}_S(f) = \frac{1}{|S|} \sum_{i \in [|S|]} \left| \max_{x':x_i \to x'} f(x') - y_i \right|.$$

The strategic empirical loss is a natural quantity to consider, since the expectation of $\widehat{\ell}_S(f)$ over $S$ is precisely the strategic population loss $\widehat{\ell}_{\mathcal{D}}(f)$, which we aim to minimize. Given the notion of strategic empirical loss, IA ERM simply finds any classifier $f$ in the hypothesis class $\mathcal{H}$ which minimizes $\widehat{\ell}_S(f)$, i.e.,

$$f \in \operatorname{argmin}_{f' \in \mathcal{H}} \widehat{\ell}_S(f').$$

We now analyze the generalization behavior of IA ERM. Recall that in the classical PAC setting without strategic manipulation, the sample complexity of ERM depends on the VC dimension of the hypothesis class $\mathcal{H}$. Here, with strategic manipulation, it appears that the VC dimension of $\mathcal{H}$ is no longer the right capacity measure to consider. Instead of $\mathcal{H}$, we consider the *effective* hypothesis class $\widehat{\mathcal{H}}$, defined below.

**Definition 2.2** (Effective Classifier / Hypothesis Class)**.** Fixing a feature space $\mathcal{X}$ and a reporting structure $\to$, for each classifier $f \subseteq \mathcal{X}$, the *effective classifier* $\widehat{f}$ consists of the set of features to which $f$ effectively assigns label 1, i.e.,

$$\widehat{f} = \{x \in \mathcal{X} \mid \exists x' : x \to x', f(x') = 1\}.$$

Fixing a hypothesis class $\mathcal{H}$, the effective hypothesis class $\widehat{\mathcal{H}}$ consists of the effective classifier of every classifier in $\mathcal{H}$, i.e., $\widehat{\mathcal{H}} = \{\widehat{f} \mid f \in \mathcal{H}\}$.

One may intuitively interpret the effective hypothesis class in the following way. Any classifier $f$ in the presence of strategic manipulation is equivalent to the effective classifier $\widehat{f}$ without strategic manipulation. Therefore, by considering the effective hypothesis class $\widehat{\mathcal{H}}$, we can effectively ignore strategic manipulation, and reduce the generalization analysis to that in the classical PAC setting, i.e., Theorem 2.1. This gives us the following theorem.

**Theorem 2.2** (Sample Complexity of IA ERM)**.** *Fix a feature space $\mathcal{X}$, a population distribution $\mathcal{D}$, a reporting structure $\to$, and a hypothesis class $\mathcal{H}$. For any $\varepsilon > 0$, $\delta > 0$, given*

$$m = O\left((d_{\mathrm{VC}}(\widehat{\mathcal{H}}) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples $S \sim \mathcal{D}^m$, with probability at least $1 - \delta$, any classifier $f \in \mathcal{H}$ minimizing the strategic empirical loss on $S$, i.e., $f \in \operatorname{argmin}_{f' \in \mathcal{H}} \widehat{\ell}_S(f')$, has strategic population loss at most $\widehat{\ell}_{\mathcal{D}}(f) \le \widehat{\ell}_{\mathcal{D}}(\mathcal{H}) + \varepsilon$. Moreover, finding any classifier achieving the same relative loss $\varepsilon$ and failure probability $\delta$ requires $\Omega\left((d_{\mathrm{VC}}(\widehat{\mathcal{H}}) + \log(1/\delta))/\varepsilon^2\right)$ iid samples.*

The proof of Theorem 2.2, as well as all other proofs, are in Section 2.6. In words, the above theorem says that IA ERM finds a classifier whose strategic population loss is close to the best classifier in the hypothesis class $\mathcal{H}$, and the sample complexity of finding such a classifier is determined by the VC dimension of the effective hypothesis class $\widehat{\mathcal{H}}$, rather than that of $\mathcal{H}$ itself.

We make a few remarks regarding IA ERM. First, when the reporting structure $\to$ is trivial, i.e., a point with feature $x$ can only report $x$ itself, then the effective class $\widehat{\mathcal{H}} = \mathcal{H}$, and IA ERM reduces to vanilla ERM. In such cases, Theorem 2.2 reduces precisely to the classical Theorem 2.1. Second, in general, the VC dimension $d_{\mathrm{VC}}(\widehat{\mathcal{H}})$ of the effective class $\widehat{\mathcal{H}}$ may be smaller or larger than that of $\mathcal{H}$. Below we give two examples, showing that either of the two quantities can be arbitrarily larger than the other.

We first give an example where $d_{\mathrm{VC}}(\mathcal{H}) = \infty$ and $d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = 1$. Consider the example discussed in the introduction. There, the feature space $\mathcal{X} = \mathbb{R}_+ = [0, \infty)$, and the reporting structure $\to$ satisfies for any $x_1, x_2 \in \mathbb{R}_+$,

$$x_1 \to x_2 \iff x_1 \ge x_2.$$

Consider $\mathcal{H} = 2^{\mathbb{R}_+}$. Clearly $d_{\mathrm{VC}}(\mathcal{H}) = \infty$, since any $S \subseteq \mathbb{R}_+$ is shattered by $\mathcal{H}$. On the other hand, for any $f \in \mathcal{H}$, $\widehat{f}$ is essentially determined by a threshold $\theta_f$, where

$$\theta_f = \inf\{x \mid f(x) = 1\}.$$

$\widehat{f}$ is then defined by

$$\widehat{f}(x) = \begin{cases} 0, & x < \theta_f \\ 1, & x > \theta_f \\ f(x), & x = \theta_f. \end{cases}$$

In other words, $\widehat{\mathcal{H}}$ is the class of threshold classifiers over $\mathbb{R}_+$. It is well-known that $d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = 1$.

Now we show that $d_{\mathrm{VC}}(\widehat{\mathcal{H}})$ can be arbitrarily larger than $d_{\mathrm{VC}}(\mathcal{H})$. Let the feature space be $\mathcal{X} = \mathbb{N}$, and $\mathcal{H} = \{\{i\} \mid i \in \mathbb{N}\}$, i.e., the set of all singletons. It is clear that $d_{\mathrm{VC}}(\mathcal{H}) = 1$. Now for

any $d \in \mathbb{N}$, we construct $\to$ such that $d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = d$.[6] Let $\to$ be such that (1) $i \to i$ for any $i \in \mathbb{N}$, and (2) for any $i \in \{d, \ldots, d + 2^d - 1\}$ and $j \in \{0, \ldots, d-1\}$, $j \to i$ if the $(j+1)$-th digit in the binary representation of $i - d$ is 1. Now we argue that $\widehat{\mathcal{H}}$ shatters $\{0, \ldots, d-1\}$. In fact, any subset $S \subseteq \{0, \ldots, d-1\}$ can be viewed as the binary representation of an integer $i_S$ in $\{0, \ldots, 2^d - 1\}$, where the $(j+1)$-th digit of $i_S$ is 1 iff $j \in S$. Consider the classifier $f_S = \{i_S + d\} \in \mathcal{H}$. Clearly $\widehat{f_S} \cap \{0, d-1\} = S$, since precisely the points in $S$ can report $i_S + d$, which is the only point assigned label 1 by $f_S$.

## 2.4 Incentive-Compatible Empirical Risk Minimization

The IA ERM principle, together with Theorem 2.2, provides a rather general solution for PAC learning in the presence of strategic manipulation. Still, one may wonder whether it is possible to achieve more desirable properties, e.g., enhanced robustness against strategic manipulation and better generalization guarantees, by further refining / regularizing IA ERM. In this section, we propose *incentive compatibility as a means of regularization*, which leads to the incentive-compatible ERM principle (henceforce IC ERM).

We first present the IC ERM principle and provide a general analysis of its sample complexity in Section 2.4.1. In particular, we show that IC ERM generalizes no worse than IA ERM when applied to the same hypothesis class. In Section 2.4.2, we consider a special case of IC ERM, *free* IC ERM, where the hypothesis class $\mathcal{H}$ is implicitly all possible classifiers over the feature space $\mathcal{X}$, i.e., $\mathcal{H} = 2^{\mathcal{X}}$. We give an efficient algorithm for free IC ERM, and more importantly, we show that, somewhat surprisingly, it is still possible to derive nontrivial generalization bounds for free IC ERM. Based on the generalization analysis of free IC ERM, we provide a hypothesis-class-independent sample complexity bound for IC ERM, which further illustrates the power of incentive compatibility as a means of regularization. Finally, in Section 2.4.3, we discuss an important class of reporting structures, i.e., *transitive* reporting structures, on which IC ERM is equivalent to the more general IA ERM. Based on this equivalence, we give a hypothesis-class-independent sample complexity bound for IA ERM that applies whenever the reporting structure is transitive.

### 2.4.1 The Incentive-Compatible ERM Principle

First we introduce the notions of incentive-compatible classifiers and hypothesis classes. Incentive compatibility is a standard concept in mechanism design and straightforwardly applying it in our context results in the following definition.

**Definition 2.3** (Incentive-Compatible Classifiers / Hypothesis Classes)**.** Fixing a feature space $\mathcal{X}$ and a reporting structure $\to$, a classifier $f : \mathcal{X} \to \{0, 1\}$ is *incentive compatible* if for any $x_1, x_2 \in \mathcal{X}$,

$$x_1 \to x_2 \Longrightarrow f(x_1) \geq f(x_2).$$

In other words, no point can receive a better outcome by pretending to have a different feature. A hypothesis class $\mathcal{H}$ is incentive compatible if it contains only incentive-compatible classifiers.

[6] Similar constructions could also give $d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = \infty$.

The intuition behind the definitions is simple: when an incentive-compatible classifier is deployed, no point is motivated to strategically modify its feature, since it is impossible to obtain a better outcome by doing so. As a result, the effective classifier induced by any incentive-compatible classifier $f$ is itself, i.e., $\widehat{f} = f$, and the strategic (empirical) loss of an incentive-compatible classifier $f$ is the same as the (empirical) loss of the same classifier without strategic manipulation, i.e.,

$$\widehat{\ell}_{\mathcal{D}}(f) = \ell_{\mathcal{D}}(f) \quad \text{and} \quad \widehat{\ell}_{S}(f) = \ell_{S}(f).$$

Incentive-compatible classifiers therefore provide arguably the strongest robustness one may hope for against strategic manipulation — they eliminate strategic manipulation. The IC ERM principle presented below always finds a classifier that is incentive compatible.

**Incentive-compatible ERM.** For any hypothesis class $\mathcal{H}$, let the *incentive-compatible subclass* $\mathcal{H}^{\mathrm{IC}(\to)}$ be the largest subset of $\mathcal{H}$ that is incentive compatible under $\to$, i.e.,

$$\mathcal{H}^{\mathrm{IC}(\to)} = \{f \in \mathcal{H} \mid f \text{ is incentive compatible under } \to\}.$$

We omit the parameter $\to$ when it is clear from the context. Given a hypothesis class $\mathcal{H}$, the IC ERM principle finds any classifier $f$ in the incentive-compatible subclass $\mathcal{H}^{\mathrm{IC}}$ minimizing the empirical loss $\ell_S(f)$ on the sample set $S$, i.e.,

$$f \in \operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \ell_S(f') = \operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \widehat{\ell}_S(f').$$

We now analyze the sample complexity of IC ERM. Observe that IC ERM with hypothesis class $\mathcal{H}$ is equivalent to vanilla ERM with hypothesis class $\mathcal{H}^{\mathrm{IC}}$. Therefore, applying Theorem 2.1 to $\mathcal{H}^{\mathrm{IC}}$, we immediately obtain the following asymptotically optimal sample complexity bound for IC ERM.

**Theorem 2.3** (Sample Complexity of IC ERM)**.** *Fix a feature space $\mathcal{X}$, a population distribution $\mathcal{D}$, a reporting structure $\to$, and a hypothesis class $\mathcal{H}$. For any $\varepsilon > 0$, $\delta > 0$, given*

$$m = O\left((d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples $S \sim \mathcal{D}^m$, with probability at least $1 - \delta$, any classifier $f \in \mathcal{H}^{\mathrm{IC}}$ minimizing the empirical loss on $S$, i.e., $f \in \operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \ell_S(f')$, has strategic population loss at most $\widehat{\ell}_{\mathcal{D}}(f) \leq \widehat{\ell}_{\mathcal{D}}(\mathcal{H}^{\mathrm{IC}}) + \varepsilon$. Moreover, finding any incentive-compatible classifier within $\mathcal{H}$ achieving the same relative loss $\varepsilon$ and failure probability $\delta$ requires $\Omega\left((d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) + \log(1/\delta))/\varepsilon^2\right)$ iid samples.*

Theorem 2.3 is but a direct application of the classical Theorem 2.1. To obtain further insights into the sample complexity of IC ERM, we need to take a closer look at the VC dimension of the incentive-compatible subclass $\mathcal{H}^{\mathrm{IC}}$, which dictates the sample complexity. As discussed above, IC ERM can be viewed as a regularized version of IA ERM. Below we formalize this intuition, by showing that the sample complexity of IC ERM is in fact no larger than that of IA ERM, or that of vanilla ERM, when applied to the same hypothesis class $\mathcal{H}$. This is done in the following claim via upper bounding the VC dimension of $\mathcal{H}^{\mathrm{IC}}$ by that of $\widehat{\mathcal{H}}$, as well as that of $\mathcal{H}$.

**Proposition 2.1.** *Fixing a feature space $\mathcal{X}$ and a reporting structure $\to$, for any hypothesis class $\mathcal{H}$ over $\mathcal{X}$,*

$$\mathcal{H}^{\mathrm{IC}} \subseteq \mathcal{H} \cap \widehat{\mathcal{H}}.$$

*As a result,*

$$d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) \leq \min\{d_{\mathrm{VC}}(\mathcal{H}), d_{\mathrm{VC}}(\widehat{\mathcal{H}})\}.$$

In many natural scenarios, the VC dimension of $\mathcal{H}^{\mathrm{IC}}$ is significantly smaller than $d_{\mathrm{VC}}(\widehat{\mathcal{H}})$. Below we give an example where $d_{\mathrm{VC}}(\mathcal{H}) = \infty$ and $d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) = 1$. Consider again the introductory example, where $\mathcal{X} = \mathbb{R}_+$. As argued above, when $\mathcal{H} = 2^{\mathbb{R}_+}$, $\widehat{\mathcal{H}}$ is all threshold classifiers, and $d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = 1$. On the other hand, by Proposition 2.1, $\mathcal{H}^{\mathrm{IC}} \subseteq \widehat{\mathcal{H}}$. In fact, one may show that every classifier in $\widehat{\mathcal{H}}$ is incentive-compatible, and $\mathcal{H}^{\mathrm{IC}} = \widehat{\mathcal{H}}$. As a result, $d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) = d_{\mathrm{VC}}(\widehat{\mathcal{H}}) = 1$.

So, in addition to the highly desirable property of incentive compatibility, IC ERM also has at most the same, and often much better sample complexity than IA ERM, which translates to better generalization fixing the number of samples. We also remark that IC ERM finds an approximately optimal classifier in the incentive-compatible subclass $\mathcal{H}^{\mathrm{IC}}$, which may have a worse approximation error than $\mathcal{H}$. In other words, to achieve incentive compatibility, one in general has to sacrifice some accuracy in the form of a larger approximation error. We will see more desirable properties of IC ERM in the rest of this section.

### 2.4.2 Free IC ERM and Generalization from Incentive Compatibility

We now consider free IC ERM, which is a special case of IC ERM where the hypothesis class consists of all possible classifiers over $\mathcal{X}$, i.e., $\mathcal{H} = \mathcal{H}_0 = 2^{\mathcal{X}}$. At first glance this may appear senseless — in the classical PAC setting without strategic manipulation, no (nontrivial) generalization is possible if nothing is known about the ground truth a priori, i.e., when the hypothesis class consists of all possible classifiers. However, as we show below, the reporting structure, together with the requirement of incentive compatibility, in fact induces nontrivial structure over the incentive-compatible subclass $\mathcal{H}_0^{\mathrm{IC}}$, which allows nontrivial sample complexity and generalization bounds. These structures are captured by the following complexity measure, which we term the *intrinsic VC dimension*.

**Definition 2.4** (Intrinsic VC Dimension)**.** Fix a reporting structure $\to$ over a feature space $\mathcal{X}$. For any $x, x' \in \mathcal{X}$, we say $x$ can *reach* $x'$ (denoted $x \Rightarrow x'$) if there exists a sequence of features $x_1, \ldots, x_k$ for some integer $k > 0$, such that $x_1 = x$, $x_k = x'$, and for any $i \in [k-1]$, $x_i \to x_{i+1}$. A set of features $S \subseteq \mathcal{X}$ is *independent* if for any $x, x' \in S$ where $x \neq x'$, $x$ cannot reach $x'$. The intrinsic VC dimension of $\to$ over $\mathcal{X}$, $d_{\mathrm{VC}}(\mathcal{X}, \to)$, is the cardinality of any maximum subset of $\mathcal{X}$ that is independent, i.e.,

$$d_{\mathrm{VC}}(\mathcal{X}, \to) = \\ \max\{|S| \mid S \subseteq \mathcal{X} : \nexists\, x, x' \in S \text{ s.t. } x \neq x' \text{ and } x \Rightarrow x'\}.$$

It turns out that the intrinsic VC dimension of the reporting structure is precisely the VC dimension of the incentive-compatible subclass $\mathcal{H}_0^{\mathrm{IC}}$ of the null hypothesis class $\mathcal{H}_0$, as formalized in the following proposition.

**Proposition 2.2** (VC Dimension of Null Hypothesis Class)**.** *For any feature space $\mathcal{X}$ and reporting structure $\to$ over $\mathcal{X}$,*

$$d_{\mathrm{VC}}(\mathcal{X}, \to) = d_{\mathrm{VC}}(\mathcal{H}_0^{\mathrm{IC}(\to)}),$$

*where $\mathcal{H}_0 = 2^{\mathcal{X}}$ is the null hypothesis class.*

Note that for any hypothesis class $\mathcal{H} \subseteq 2^{\mathcal{X}} = \mathcal{H}_0$ over $\mathcal{X}$, the incentive-compatible subclass of $\mathcal{H}$ is a subclass of that of the null hypothesis class $\mathcal{H}_0$, i.e., $\mathcal{H}^{\mathrm{IC}} \subseteq \mathcal{H}_0^{\mathrm{IC}}$. As a result, we always have

$$d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) \leq d_{\mathrm{VC}}(\mathcal{H}_0^{\mathrm{IC}}) = d_{\mathrm{VC}}(\mathcal{X}, \to).$$

This, together with Theorem 2.3, immediately implies the following hypothesis-class-independent sample complexity bound for IC ERM, which, in particular, applies to free IC ERM.

**Theorem 2.4** (Hypothesis-Class-Independent Sample Complexity Bound for IC ERM)**.** *Fix a feature space $\mathcal{X}$, a population distribution $\mathcal{D}$, a reporting structure $\to$, and a hypothesis class $\mathcal{H}$. For any $\varepsilon > 0$, $\delta > 0$, given*

$$m = O\left((d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples $S \sim \mathcal{D}^m$, with probability at least $1 - \delta$, any classifier $f \in \mathcal{H}^{\mathrm{IC}}$ minimizing the empirical loss on $S$, i.e., $f \in \operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \ell_S(f')$, has strategic population loss $\widehat{\ell}_{\mathcal{D}}(f) \leq \widehat{\ell}_{\mathcal{D}}(\mathcal{H}^{\mathrm{IC}}) + \varepsilon$. Moreover, when the hypothesis class $\mathcal{H}$ is the null hypothesis class $\mathcal{H}_0$, finding any incentive-compatible classifier achieving the same relative loss $\varepsilon$ and failure probability $\delta$ requires*

$$\Omega\left((d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples.*

We also present an efficient algorithm for free IC ERM in Section 2.5.

### 2.4.3 Transitive Reporting and the Revelation Principle

Finally, we investigate an important family of reporting structures, transitive reporting structures. It turns out that with transitivity, the so-called revelation principle from mechanism design holds: if one accounts for strategic reporting, then without loss of generality one may focus on incentive-compatible classifiers. That is, IC ERM is as general as IA ERM. As a result, with transitivity, the hypothesis-class-independent sample complexity bound extends to IA ERM applied to any hypothesis class. We first give the formal definition of transitive reporting structures, which is essentially the same as that of transitive binary relations.

**Definition 2.5** (Transitive Reporting Structures)**.** A reporting structure $\to$ over $\mathcal{X}$ is *transitive* if for any $x_1, x_2, x_3 \in \mathcal{X}$,

$$(x_1 \to x_2 \text{ and } x_2 \to x_3) \implies x_1 \to x_3.$$

One example of transitive reporting structures is the example from the introduction, where $\mathcal{X} = \mathbb{R}_+$, and $\to$ is precisely the same as $\geq$, which is clearly transitive. It turns out that transitivity of reporting structures is equivalent to the revelation principle holding, which roughly says that any classifier is equivalent to a (possibly different) incentive-compatible classifier. Formally:

**Proposition 2.3** (Revelation Principle for PAC Learning)**.** *The following is true if and only if the reporting structure is transitive: for any classifier $f : \mathcal{X} \to \{0,1\}$, the effective classifier $\widehat{f}$ is incentive compatible, and as a result, for any hypothesis class $\mathcal{H}$, the effective class $\widehat{\mathcal{H}}$ is incentive-compatible (i.e., $(\widehat{\mathcal{H}})^{\mathrm{IC}} = \widehat{\mathcal{H}}$).*

In light of Proposition 2.3, when the reporting structure is transitive, for any $\mathcal{H}$, IA ERM with hypothesis class $\mathcal{H}$ is equivalent to IC ERM with hypothesis class $\widehat{\mathcal{H}}$, in the sense that they yield the same effective classifier given any sample set, with the same sample complexity. This gives us a way to translate the hypothesis-class-independent sample complexity bound for IC ERM to IA ERM, and obtain the following theorem.

**Theorem 2.5** (Hypothesis-Class-Independent Sample Complexity Bound for IA ERM)**.** *Fix a feature space $\mathcal{X}$, a population distribution $\mathcal{D}$, a transitive reporting structure $\to$, and a hypothesis class $\mathcal{H}$. For any $\varepsilon > 0$, $\delta > 0$, given*

$$m = O\left((d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta))/\varepsilon^2\right)$$

*iid samples $S \sim \mathcal{D}^m$, with probability at least $1-\delta$, any classifier $f \in \mathcal{H}$ minimizing the strategic empirical loss on $S$, i.e., $f \in \operatorname{argmin}_{f' \in \mathcal{H}} \widehat{\ell}_S(f')$, has strategic population loss $\widehat{\ell}_{\mathcal{D}}(f) \le \widehat{\ell}_{\mathcal{D}}(\mathcal{H}) + \varepsilon$. Moreover, when the hypothesis class $\mathcal{H}$ is the null hypothesis class $\mathcal{H}_0$, finding any classifier achieving the same relative loss $\varepsilon$ and failure probability $\delta$ requires $\Omega\left((d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta))/\varepsilon^2\right)$ iid samples.*

To develop intuition, consider again the introductory example. As discussed above, there, $\mathcal{X} = \mathbb{R}_+$ and the reporting structure $\to = \ge$ is transitive. Moreover, the intrinsic VC dimension of $\to$ is $d_{\mathrm{VC}}(\mathcal{X}, \to) = 1$. This is because any singleton set is independent, and for any $x_1 \neq x_2$, either $x_1 < x_2$ or $x_2 < x_1$, so $\{x_1, x_2\}$ cannot be independent. On the other hand, as argued above, any classifier $f \in 2^{\mathcal{X}}$ effectively implements a threshold $\theta_f$, where $f(x) = 1$ iff $x \ge \theta_f$ or $x > \theta_f$. It is well-known (see, e.g., [93]) that $\Theta(\log(1/\delta)/\varepsilon^2)$ samples suffice to learn such a threshold with relative loss $\varepsilon$ and failure probability $\delta$, which coincides with the above sample complexity bound based on the intrinsic VC dimension.

## 2.5 Algorithm for Free IC ERM

In this section we present an efficient algorithm, Algorithm 2.1, for free IC ERM. We show below that Algorithm 2.1 does compute an empirical risk minimizer among all incentive-compatible classifiers.

**Theorem 2.6.** *Algorithm 2.1 finds a classifier $f$ which satisfies*

$$f \in \operatorname{argmin}_{f' \in \mathcal{H}_0^{\mathrm{IC}}} \ell_S(f').$$

## 2.6 Omitted Proofs

*Proof of Theorem 2.2.* First observe that for any classifier $f \in 2^{\mathcal{X}}$, the strategic population loss $\widehat{\ell}_{\mathcal{D}}(f) = \ell_{\mathcal{D}}(\widehat{f})$, and the strategic empirical loss $\widehat{\ell}_S(f) = \ell_S(\widehat{f})$. The plan is to apply Theorem 2.1

**Algorithm 2.1:** Algorithm for Free IC ERM

**Input:** A reporting structure $\to$ over feature space $\mathcal{X}$, a sample set $S = \{(x_i, y_i)\}_i \sim \mathcal{D}^m$ of size $m$.

**Output:** An incentive-compatible classifier $f$ minimizing the empirical loss $\ell_S(f)$ on $S$.

Let $G = (V, E)$ be a capacitated directed graph, where $V = \{x_i\}_{i\in[m]} \cup \{s, t\}$ and $E = \emptyset$.
**for** $1 \le i < j \le m$ **do**
  **if** $x_i \to x_j$ **then**
    Let $E \leftarrow E \cup \{(x_j, x_i, \infty)\}$, i.e., add an edge from $x_j$ to $x_i$ with capacity $\infty$.
  **end if**
**end for**
**for** $i \in [m]$ **do**
  **if** $y_i = 0$ **then**
    Let $E \leftarrow E \cup \{(x_i, t, 1)\}$, i.e., add an edge from $x_i$ to $t$ with capacity $1$.
  **else**
    Let $E \leftarrow E \cup \{(s, x_i, 1)\}$, i.e., add an edge from $s$ to $x_i$ with capacity $1$.
  **end if**
**end for**
Compute an $s$-$t$ mincut $(C, \overline{C})$ on $G$, where $C$ is the set of vertices on the $s$ side of the cut.
Let $f$ be such that for any $x \in \mathcal{X}$, $f(x) = 1$ iff there exists $x' \in C \setminus \{s\}$ where $x \Rightarrow x'$; return $f$.

to the effective hypothesis class $\widehat{\mathcal{H}}$. Let

$$f \in \operatorname{argmin}_{f'\in\mathcal{H}} \widehat{\ell}_S(f') = \operatorname{argmin}_{f'\in\mathcal{H}} \ell_S(\widehat{f}').$$

We then have

$$\widehat{f} \in \operatorname{argmin}_{f'\in\widehat{\mathcal{H}}} \ell_S(f').$$

That is, $\widehat{f}$ is a minimizer of $\ell_S$ in $\widehat{\mathcal{H}}$. By Theorem 2.1, with $m$ samples, with probability at least $1 - \delta$,

$$\ell_{\mathcal{D}}(\widehat{f}) \le \ell_{\mathcal{D}}(\widehat{\mathcal{H}}) + \varepsilon = \widehat{\ell}_{\mathcal{D}}(\mathcal{H}).$$

On the other hand, Theorem 2.1 states that finding such an $\widehat{f}$ in $\widehat{\mathcal{H}}$ with relative loss $\varepsilon$ and failure probability $\delta$ requires asymptotically the same number of samples. This conlcudes the proof of the theorem. □

*Proof of Theorem 2.3.* Theorem 2.3 is a direct corollary of Theorem 2.1 (or Theorem 2.2 which is more general). Applying Theorem 2.1 with hypothesis class $\mathcal{H}^{\mathrm{IC}}$, we immediately obtain that with $m$ samples, for any

$$f \in \operatorname{argmin}_{f'\in\mathcal{H}^{\mathrm{IC}}} \ell_S(f'),$$

with probability $1 - \delta$,

$$\ell_{\mathcal{D}}(f) \le \ell_{\mathcal{D}}(\mathcal{H}^{\mathrm{IC}}) + \varepsilon.$$

And moreover, the number of samples is asymptotically tight. On the other hand, since $\mathcal{H}^{\mathrm{IC}}$ is incentive-compatible,

$$\operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \ell_S(f') = \operatorname{argmin}_{f' \in \mathcal{H}^{\mathrm{IC}}} \widehat{\ell}_S(f'),$$

and

$$\widehat{\ell}_{\mathcal{D}}(f) \leq \widehat{\ell}_{\mathcal{D}}(\mathcal{H}^{\mathrm{IC}}) + \varepsilon.$$

This concludes the proof. □

*Proof of Proposition 2.1.* For any $f \in \mathcal{H}^{\mathrm{IC}}$, since $f$ is incentive-compatible, we have

$$f = \widehat{f} \in \widehat{\mathcal{H}},$$

and therefore $\mathcal{H}^{\mathrm{IC}} \subseteq \widehat{\mathcal{H}}$. On the other hand, by definition, $\mathcal{H}^{\mathrm{IC}} \subseteq \mathcal{H}$. This implies that $\mathcal{H}^{\mathrm{IC}} \subseteq \mathcal{H} \cap \widehat{\mathcal{H}}$. □

*Proof of Proposition 2.2.* First we show $d_{\mathrm{VC}}(\mathcal{H}_0^{\mathrm{IC}}) \geq d_{\mathrm{VC}}(\mathcal{X}, \to)$. Let $S \subseteq \mathcal{X}$ be an independent subset of $\mathcal{X}$ with cardinality $d_{\mathrm{VC}}(\mathcal{X}, \to)$. Such a subset exists by the definition of $d_{\mathrm{VC}}(\mathcal{X}, \to)$. We argue that $S$ can be shattered by $\mathcal{H}_0^{\mathrm{IC}}$. For any $T \subseteq S$, we construct a classifier $f_T \in \mathcal{H}_0^{\mathrm{IC}}$ such that for any $x \in S$, $f_T(x) = 1 \iff x \in T$. Let $f_T$ be such that

$$f_T(x) = \begin{cases} 1, & \text{if there exists } x' \in T : x \Rightarrow x' \\ 0, & \text{otherwise.} \end{cases}$$

We only need to check that $f_T$ is incentive-compatible. Suppose otherwise, i.e., there exist $x_1, x_2 \in \mathcal{X}$, such that $x_1 \to x_2$, $f_T(x_1) = 0$, and $f_T(x_2) = 1$. It must be the case that for some $x_3 \in T$, such that $x_2 \Rightarrow x_3$. Then, by definition, we have $x_1 \Rightarrow x_3$, and therefore it should be the case that $f_T(x_1) = 1$, a contradiction.

Now we show $d_{\mathrm{VC}}(\mathcal{H}_0^{\mathrm{IC}}) \leq d_{\mathrm{VC}}(\mathcal{X}, \to)$. That is, for any subset $S \subseteq \mathcal{X}$ where $|S| > d_{\mathrm{VC}}(\mathcal{X}, \to)$, $S$ cannot be shattered by $\mathcal{H}_0^{\mathrm{IC}}$. By the definition of $d_{\mathrm{VC}}(\mathcal{X}, \to)$, $S$ cannot be independent. Let $x, x' \in S$ be such that $x \Rightarrow x'$. Furthermore, let

$$x = x_1, x_2, \ldots, x_{k-1}, x_k = x' \in \mathcal{X}$$

be a sequence through which $x$ can reach $x'$, i.e., for any $i \in [k-1]$, $x_i \to x_{i+1}$. We show that for any incentive-compatible $f$, it cannot be the case that $f(x) = 0$ and $f(x') = 1$. Suppose otherwise. Let $t \in [k-1]$ be the largest integer such that $f(x_t) = 0$. $t$ exists since $f(x_1) = 0$ and $f(x_k) = 1$. Then we have $x_t \to x_{t+1}$, but $f(x_t) = 0 < 1 = f(x_{t+1})$, a contradiction. This concludes the proof of the proposition. □

*Proof of Theorem 2.4.* The theorem is a direct corollary of Theorem 2.3 and Proposition 2.2. Observe that $\mathcal{H}^{\mathrm{IC}} \subseteq \mathcal{H}_0^{\mathrm{IC}}$, and $d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) \leq d_{\mathrm{VC}}(\mathcal{H}_0^{\mathrm{IC}}) = d_{\mathrm{VC}}(\mathcal{X}, \to)$. Applying Theorem 2.3, the number of samples required for IC ERM on $\mathcal{H}$ is

$$O\left(\frac{d_{\mathrm{VC}}(\mathcal{H}^{\mathrm{IC}}) + \log(1/\delta)}{\varepsilon^2}\right) = O\left(\frac{d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta)}{\varepsilon^2}\right).$$

This concludes the proof. □

*Proof of Proposition 2.3.* Fix any $f \in \mathcal{H}$, we show $\widehat{f}$ is incentive-compatible. First observe that since $\to$ is transitive,

$$\widehat{f}(x) = 1 \iff \exists x' : x \to x' \text{ and } f(x') = 1 \iff \exists x' : x \Rightarrow x' \text{ and } f(x') = 1.$$

For any $x_1, x_2 \in \mathcal{X}$ where $x_1 \to x_2$, if $\widehat{f}(x_2) = 1$, then there exists $x'$ such that $x_2 \Rightarrow x'$ and $f(x') = 1$. Then since $x_1 \to x_2$, we have $x_1 \Rightarrow x'$, and as a result, $\widehat{f}(x_1) = 1$. This immediately implies the proposition. □

*Proof of Theorem 2.5.* The theorem is a corollary of Theorem 2.4 and Proposition 2.3. By Proposition 2.3, when $\to$ is transitive, IA ERM with hypothesis $\mathcal{H}$ is equivalent to IC ERM with hypothesis $\widehat{\mathcal{H}}$. By Theorem 2.4, the number of samples required for the latter is

$$O\left(\frac{d_{\mathrm{VC}}(\mathcal{X}, \to) + \log(1/\delta)}{\varepsilon^2}\right).$$

This concludes the proof. □

*Proof of Theorem 2.6.* First observe that given an incentive-compatible classifier $f|_S$ restricted to $\{x_i\}_i$, one can always extend the classifier by assigning label 1 to $x \in \mathcal{X}$ iff there exists $x' \in \{x_i\}_i$ where $f|_S(x') = 1$ and $x \Rightarrow x'$. Such an extension assigns label one only if incentive-compatibility is violated otherwise.

Given the above observation, we only need to show that $f$ minimizes the empirical loss among incentive-compatible classifiers restricted to the sample set $S$. We argue that each incentive-compatible classifier $f' : \{x_i\}_i \to \{0, 1\}$ corresponds bijjectively to a finite capacity $s$-$t$ cut in the graph $G$ constructed in Algorithm 2.1. Recall that a classifier $f'$ is incentive-compatible (restricted to $\{x_i\}_i$) iff for any $i, j \in [m]$,

$$x_i \to x_j \Longrightarrow f(x_i) \geq f(x_j).$$

Consider the cut $(C', \overline{C'})$ corresponding to $f'$ defined such that $x_i \in C' \iff f'(x_i) = 1$. Per the construction in Algorithm 2.1, $x_i \to x_j$ iff there is an edge from $x_j$ to $x_i$ with infinite capacity, and $f'(x_i) < f'(x_j)$ iff $x_i \notin C'$ and $x_j \in C'$. The condition for $f'$ being incentive-compatible is therefore equivalent to: no infinite capacity edge is cut by $C'$. In other words, $C'$ has finite capacity.

Now since $f$ found by the algorithm corresponds to a min-cut, it has to be incentive-compatible. We show below that $f$ also minimizes the empirical loss on $S$. We rewrite the empirical loss of $f'$ in the following way.

$$\begin{aligned}
\ell_S(f') &= \frac{1}{m} \sum_{i \in [m]} |f'(x_i) - y_i| \\
&= \frac{1}{m} \sum_{i \in [m] : y_i = 0} f'(x_i) + \frac{1}{m} \sum_{i \in [m] : y_i = 1} (1 - f'(x_i)) \\
&= \frac{1}{m} \left( \sum_{i \in [m] : y_i = 0} \mathbb{I}[x_i \in C'] + \sum_{i \in [m] : y_i = 1} \mathbb{I}[x_i \notin C'] \right).
\end{aligned}$$

Observe that the last line multiplied by $m$ is exactly the capacity of $C'$, since for each $i$ where $y_i = 0$, $x_i \in C'$ iff the edge from $s$ to $x_i$ with capacity $1$ is cut, and for each $i$ where $y_i = 1$, $x_i \notin C'$ iff the edge from $x_i$ to $t$ with capacity $1$ is cut. Therefore, minimizing the capacity of the cut is equivalent to minimizing the empirical loss of $f'$ on $S$. We conclude that $f$ found by Algorithm 2.1 is in fact an incentive-compatible classifier with minimum empirical loss on $S$. $\square$

# Chapter 3

# Automated Mechanism Design for Classification with Partial Verification

## 3.1 Introduction

As discussed in previous chapters, agents are often *classified* into a variety of categories, some more desirable than others. Loan applicants might be classified in various categories of risk, determining the interest they would have to pay. University applicants may be classified into categories such as "rejected," "wait list," "regular accept," and "accept with honors scholarship." Meanwhile universities might themselves be classified into categories such as "most competitive," "highly competitive," etc. In Chapter 2, we have primarily focused on the statistical aspect of such classification problems, i.e., how well a classifier trained on a few samples *generalizes* to the entire population. In this chapter, we turn to the computational aspect of the problem. In line with the language of *mechanism design* (often considered part of *game theory*), we assume that each agent (i.e., the entity being classified) has a *type*, corresponding to the agent's true financial situation, ability as a student, or competitiveness as a university. This type is information that is private to the agent. In most applications of mechanism design, the type encodes the agent's *preferences*. For example, in an auction, an agent's type is how much he values the outcome where he wins the auction. In contrast, in our setting, the type does not encode the agent's preferences: in the examples above, typically any agent has the same preferences over outcomes, regardless of the agent's true type. Instead, the type is relevant to the objective function of the *principal* (the entity doing the classification), who wants to classify the agents into a class that fits their type.

Often, in mechanism design, it is assumed that an agent of any type can report any other type (e.g., bid any value in an auction), and outcomes are based on these reports. Under this assumption, our problem would be hopeless: every agent would always simply report whatever type gives the most favorable outcome, so we could not at all distinguish agents based on their true type. But in our context this assumption is not sensible: while an agent may be able to take some actions that affect how its financial situation appears, it will generally not be possible for a person in significant debt and without a job to successfully imitate a wealthy person with a secure career. This brings us into the less commonly studied domain of *mechanism design with partial verification* [51, 103], in

which not every type can misreport every other type. That is, each type has certain other types that it can misreport. A standard example in this literature is that it is possible to have arrived later than one really did, but not possible to have arrived earlier. (In that case, the arrival time is the type.) In this chapter, however, we are interested in more complex misreporting (in)abilities.

What determines which types can misreport (i.e., successfully imitate) which other types? This is generally specific to the setting at hand. Zhang et al. [114] consider settings in which different types produce "samples" (e.g., timely payments, grades, admissions rates, ...) according to different distributions. They characterize which types can distinguish themselves from which other types in the long run, in a model in which agents can either (1) manipulate these samples before they are submitted to the principal, by either withholding transforming some of them in limited ways, or (2) choose the number of costly samples to generate [113, 114, 116]. Such sample-based settings are discussed in more detail later in Chapters 5, 6, and 7. In this chapter, we will take as given which types can misreport which other types; this relation may result from applying the above characterization result, or from some other model.

Our goal is: given the misreporting relation, agents' preferences, and the principal's objective, can we efficiently compute the optimal (single-agent) mechanism/classifier, which assigns each report to an outcome/class? This is a problem in *automated mechanism design* [24, 25], where the goal is to compute the optimal mechanism for the specific setting (outcome space, utility and objective functions, type distribution, ...) at hand. Quite a bit is already known about the complexity of the automated mechanism design problem, and with partial verification, the problem is known to become even harder [5, 64, 65, 103]. The structural advantage that we have here is that, unlike that earlier work, we are considering settings where all types have the same preferences over outcomes. This allows us positive results that would otherwise not be available.

### 3.1.1 Our Results and Techniques

Throughout the chapter, we assume agents have *utility* functions which they seek to maximize, and the principal has a *cost* function which she seeks to minimize.

**General vs. truthful mechanisms.** We first set out to investigate the problem of automated mechanism design with partial verification in the most general sense, where there is no restriction on each type's utility function. In light of previously known hardness results, although the most general problem is unlikely to be efficiently solvable, one may still hope to identify maximally nontrivial special cases for which efficient algorithms exist. In order to determine the boundary of tractability, our first finding, Theorem 3.1, shows that when the revelation principle does not hold, it is NP-hard to find an optimal (randomized or deterministic) mechanism even if (1) there are only 2 outcomes and (2) all types share the same utility function.[1] In other words, without the revelation principle, no efficient algorithm exists even for the minimally nontrivial setting. We

[1]The *revelation principle* states that if certain conditions hold on the reporting structure, then it is without loss of generality to focus on *truthful* mechanisms, in which agents are always best off revealing their true type. We will discuss below a necessary and sufficient condition for the revelation principle to hold in our setting.

therefore focus our attention on cases where the revelation principle holds, or, put in another way, on finding optimal truthful mechanisms.

**General vs. structured utility functions.** The above result, as well as prior results on mechanism design with partial verification [5, 64, 65, 103], paints a clear picture of intractability when the revelation principle does not hold. But prior work also often suggests that this is indeed the boundary of tractability. This is in fact true if we consider optimal randomized truthful mechanisms, which can be found by solving a linear program with polynomially many variables and constraints if the number of agents is constant [24]. However, as our second finding (Theorem 3.2) shows, the case of *deterministic* mechanisms is totally different — even with 3 outcomes and single-peaked preferences over outcomes, it is still NP-hard to find an optimal deterministic truthful mechanism (significantly improving over earlier hardness results for deterministic mechanisms [24, 25]). In other words, optimal deterministic truthful mechanisms are almost always hard to find whenever types have different preferences over outcomes. This leads us to what appears to be the only nontrivial case left, i.e., where all types share the same preference over outcomes. But this case is important: as discussed above, it in fact nicely captures a number of real-world scenarios of practical importance, and will be the focus in the rest of our results.

**Efficient algorithm for deterministic mechanisms.** Our first algorithmic result (Theorem 3.3) is an efficient algorithm for finding optimal deterministic truthful mechanisms with identical preferences in the presence of partial verification. The algorithm works by building a directed capacitated graph, where each deterministic truthful mechanism corresponds bijectively to a finite-capacity $s$-$t$ cut. The algorithm then finds an $s$-$t$ min-cut in polynomial time, which corresponds to a deterministic truthful mechanism with the minimum cost.

**Condition for deterministic optimality and faster algorithm for randomized mechanisms.** We then consider randomized mechanisms. We aim to answer the following two natural questions.

- In which cases is there a gap between optimal deterministic and randomized mechanisms, and how large can this gap be?
- While LP formulations exist for optimal randomized truthful mechanisms in general, is it possible to design theoretically and/or practically faster algorithms when types share the same utility function?

The answers to these questions turn out to be closely related.

For the first question, we show that the gap in general can be arbitrarily large (Example 3.1). On the other hand, there always exists an optimal truthful mechanism that is deterministic whenever the principal's cost function is convex with respect to the common utility function (Lemma 3.1). In order to prove this, we show that without loss of generality, an optimal truthful mechanism randomizes only between two consecutive outcomes (when sorted by utility) for each type, and present a way to round any such mechanism into a deterministic truthful mechanism, preserving the cost in expectation.

For the second question, we give a positive answer, by observing that with randomization, essentially only the convex envelope of the principal's cost function matters. This implies a reduction

from finding optimal randomized mechanisms with general costs, to finding optimal randomized mechanisms with convex costs, and – via our answer to the first question (Lemma 3.1) – to finding optimal deterministic mechanisms with convex costs. As a result, finding optimal randomized truthful mechanisms is never harder than finding optimal deterministic truthful mechanisms with convex costs. Combined with our algorithm for the latter problem (Theorem 3.3), this reduction implies a theoretically and practically faster algorithm for finding optimal randomized truthful mechanisms when types share the same utility function.

**Generalizing to combinatorial costs.** With all the intuition developed so far, we then proceed to a significantly more general setting, where the principal's cost is a function of the combination of outcomes for each type, i.e., the principal's cost function is *combinatorial*. This further captures global constraints for the principal, e.g., budget or headcount constraints. We present combinatorial counterparts of essentially all our results for additive costs in Section 3.3.

## 3.2 Additive Cost over Types

Consider the classical setting of Bayesian (single-agent) mechanism design, which is as follows. The agent can have one of many possible *types*. The agent reports a type to the principal (which may not be his true type), and then the principal chooses an *outcome*. The principal does not know the type of the agent, but she has a prior probability distribution over the agent's possible types. The principal has a different cost for each combination of a type and an outcome. The goal of the principal is to design a mechanism (a mapping from reports to outcomes) to minimize her expected cost assuming the agent best-responds to (i.e., maximizes his utility under) the mechanism. The principal aims to minimize her total cost over this population of agents, which is equal to the sum of her cost over individual agents.

In this section, we focus on the traditional setting where the principal's cost is additive over types. In Section 3.3, we generalize our results to broader settings where the principal's cost function can be combinatorial (e.g., submodular) over types.

**Notation.** Let $\Theta$ be the agent's type space, and $\mathcal{O}$ the set of outcomes. Let $n = |\Theta|$ and $m = |\mathcal{O}|$ be the numbers of types and outcomes respective. Generally, we use $i \in \Theta$ to index types, and $j \in \mathcal{O}$ to index outcomes. Let $\mathbb{R}_+ = [0, \infty)$. We use $u_i : \mathcal{O} \to \mathbb{R}_+$ to denote the utility of a type $i$ agent, and $c_i : \mathcal{O} \to \mathbb{R}_+$ to denote the cost of the principal of assigning different outcomes to a type $i$ agent.

Let $R \subseteq \Theta \times \Theta$ denote all possible ways of misreporting, that is, a type $i$ agent can report type $i'$ if and only if $(i, i') \in R$. We assume each type $i$ can always report truthfully, i.e., $(i, i) \in R$. The principal specifies a (possibly randomized) mechanism $M : \Theta \to \mathcal{O}$, which maps reported types to (distributions over) outcomes. The agent then responds to maximize his expected utility under $M$.

Let $r_i$ denote the report of type $i$ when the agent best responds:

$$r_i \in \operatorname{argmax}_{i' \in \Theta, (i,i') \in R} \mathbb{E}[u_i(M(i'))].$$

Without loss of generality, the principal's cost function can be scaled so that the prior distribution over possible types is effectively uniform. The principal's cost under mechanism $M$ is then given by

$$c(M) = \sum_{i \in \Theta} \mathbb{E}\left[c_i\left(M\left(r_i\right)\right)\right]$$

where both expectations are over the randomness in $M$. Throughout the chapter, given a set $S$, we use $\Delta(S)$ to denote the set of all distributions over $S$.

### 3.2.1 Hardness without the Revelation Principle

The well-known revelation principle states that when any type can report any other type, there always exists a truthful *direct-revelation* mechanism that is optimal for the principal.[2] However, this is not true in the case of partial verification (see, e.g., [51, 65, 103]). In fact, it is known (see Theorem 4.10 of [65]) that in our setting, the revelation principle holds if and only if the reporting structure $R$ is transitive, i.e., for any types $i_1, i_2, i_3 \in \Theta$,

$$(i_1, i_2) \in R \text{ and } (i_2, i_3) \in R \implies (i_1, i_3) \in R.^3$$

We begin our investigation by presenting a hardness result, which states that when the revelation principle does not hold, it is NP-hard to find any optimal mechanism (even in the minimal nontrivial setting).

**Theorem 3.1** (NP-hardness without the Revelation Principle)**.** *When partial verification is allowed and the revelation principle does not hold, it is* NP*-hard to find an optimal (randomized or deterministic) mechanism, even if there are only* 2 *outcomes and all types share the same utility function.*

We postpone the proof of Theorem 3.1, as well as all other proofs in this section, to Section 3.6. In light of Theorem 3.1, in the rest of the chapter, we focus on finding optimal truthful direct-revelation mechanisms. That is, we consider only mechanisms $M$ where for any $(i_1, i_2) \in R$,

$$\mathbb{E}[u_{i_1}(M(i_1))] \geq \mathbb{E}[u_{i_1}(M(i_2))].$$

### 3.2.2 General vs. Structured Utility Functions

Following the convention in the literature, we assume agents always break ties by reporting truthfully. As a result, for a (possibly randomized) truthful mechanism $M$, the cost of the principal can

[2] A direct-revelation mechanism is a mechanism in which agents can only report their type, rather than sending arbitrary messages. A mechanism is truthful if it is always optimal for agents to report their true types.

[3] To get some intuition for this characterization, suppose that $(i_1, i_2) \in R$, $(i_2, i_3) \in R$, but $(i_1, i_3) \notin R$, and we would like to accept $i_2$ and $i_3$ but not $i_1$. That is, higher types are better, and each type (except for the top one) can make itself look a bit, but not much, better than it is. There is no truthful mechanism that achieves what we want: if we accept a report of $i_2$, we will end up accepting $i_1$ as well because it can misreport $i_2$. On the other hand, if we accept only $i_3$, then we get what we want, by relying on $i_2$ to non-truthfully report $i_3$ (whereas $i_1$ cannot). Hence, our goal can be achieved in a non-truthful implementation while it cannot be achieved in a truthful implementation, showing that the revelation principle does not hold in this case.

be written as

$$c(M) = \sum_{i \in \Theta} \mathbb{E}[c_i(M(i))].$$

Our first finding establishes a dichotomy between deterministic and randomized mechanisms when agents can have arbitrary utility functions. On one hand, it is known that an optimal randomized mechanism can be found in polynomial time by formulating the problem as a linear program [24]. On the other hand, finding an optimal deterministic mechanism is NP-hard even in an extremely simple setting as described below.

**Theorem 3.2** (NP-hardness with General Utility Functions)**.** *When partial verification is allowed, even when the revelation principle holds, it is* NP*-hard to find an optimal deterministic mechanism, even if there are only* 3 *outcomes and the utility functions are single-peaked (see Section 3.5 for a definition).*

Although Theorem 3.2 establishes hardness for finding optimal deterministic mechanisms in most nontrivial cases, it leaves the possibility of efficient algorithms when all types have the same utility function — which, as discussed in the introduction, is the setting we focus on in this chapter.

### 3.2.3 Finding Optimal Deterministic Mechanisms

In light of the previously mentioned hardness results, for the rest of this section, we focus on the setting where the revelation principle holds and all types have the same utility function.

We recall and simplify some notations before we state the main result of this section (Theorem 3.3). Let $u : \mathcal{O} \to \mathbb{R}_+$ be the common utility function of all types. Recall that $n = |\Theta|$ is the number of types and $m = |\mathcal{O}|$ is the number of outcomes. Let $\Theta = [n] = \{1, \ldots, n\}$. For brevity, we use $\mathcal{O} = \{o_1, \ldots, o_m\} \subseteq \mathbb{R}_+$ to encode the utility function $u$. That is, for all $j \in [m]$, $o_j \in R_+$ is the utility of the agent under the $j$-th outcome. Without loss of generality, assume $o_1 = 0$, and $o_j < o_{j+1}$ for all $j \in [m-1]$.

We give an efficient algorithm (Algorithm 3.1) for finding an optimal deterministic mechanism when partial verification is allowed. Our algorithm first builds a (capacitated) directed graph based on the principal's cost function and the reporting structure, then finds an $s$-$t$ min-cut in the graph, and then constructs a mechanism based on the found min-cut. The idea is finite-capacity cuts in the graph constructed correspond bijectively to truthful mechanisms, where the capacity is precisely the cost of the principal. In particular, we use edges with $\infty$ capacity to ensure that if one type gets an outcome, any type that can misreport the former must get at least as good an outcome. See Figure 3.1 for an illustration of Algorithm 3.1. The following theorem establishes the correctness and time complexity of Algorithm 3.1.

**Theorem 3.3** (Fast Algorithm for Finding Optimal Deterministic Mechanisms)**.** *Suppose for any $i \in [n]$ and $j \in [m]$, $c_i(o_j) \in \mathbb{N}$. Let $W = \max_{i,j} c_i(o_j)$. Algorithm 3.1 outputs an optimal deterministic truthful mechanism in time $O(T_{\mathsf{MinCut}}(mn, mn^2, W))$, where $T_{\mathsf{MinCut}}(n', m', W')$ is the time it takes to find an $s$-$t$ min-cut in a graph with $n'$ vertices, $m'$ edges, and maximum capacity $W'$.*

We note that Algorithm 3.1 only finds an optimal deterministic mechanism *subject to truthfulness* — when the revelation principle does not hold, Algorithm 3.1 may not find an unconditionally

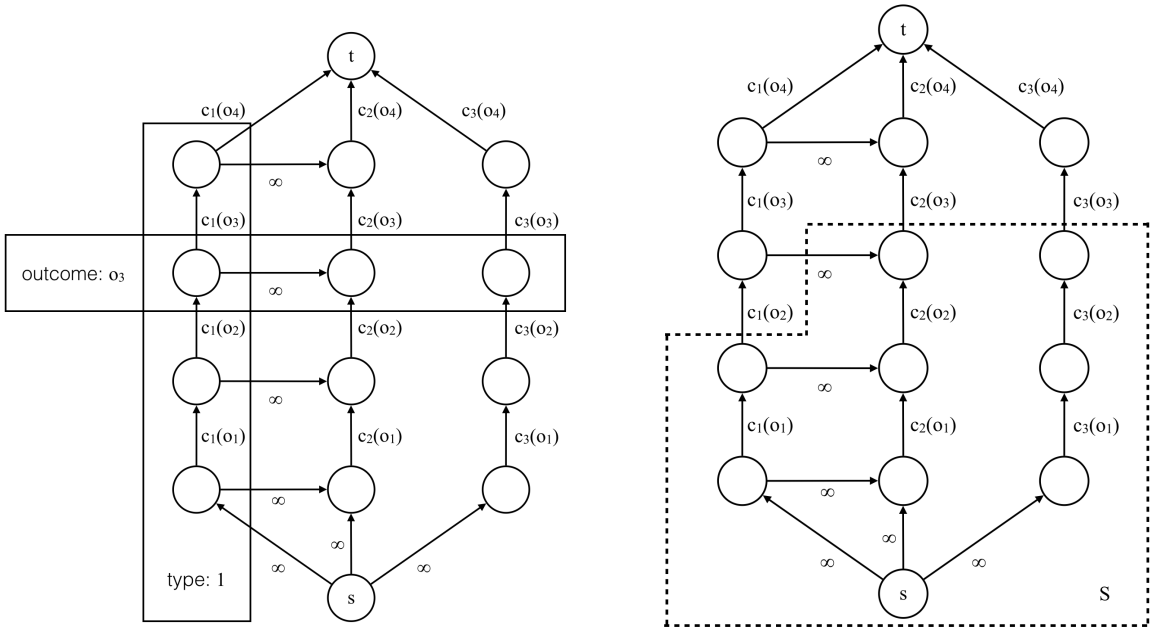


Figure 3.1: An example of the graph constructed in Algorithm 3.1. As highlighted in the left graph, each row corresponds to an outcome and each column corresponds to a type. The horizontal edges with infinite capacity correspond to the fact that type 2 can misreport as type 1. The right graph gives a possible $s$-$t$ min-cut, which corresponds to a mechanism where $M(1) = o_2$, $M(2) = (o_3)$, and $M(3) = o_3$. The horizontal edges make sure that type 1 never gets a more desirable outcome than type 2, so type 2 never misreports. The cost of the mechanism $M$ is equal to the value of the min-cut, which is $c_1(o_2) + c_2(o_3) + c_3(o_3)$.

optimal mechanism (and indeed finding that is NP-hard given Theorem 3.1). The same applies for all our algorithmic results.

### 3.2.4 Optimality of Deterministic Mechanisms with Convex Costs

In the previous subsection, we showed that when the revelation principle holds and all types have the same utility function, there is a min-cut-based algorithm (Algorithm 3.1) that finds an optimal deterministic truthful mechanism.

In this subsection, we identify an important special case where there exists an optimal truthful mechanism that is deterministic (even when randomized mechanisms are allowed). Consequently, we have an algorithm (Algorithm 3.1) for finding the optimal truthful mechanism that runs faster than solving a linear program. More importantly, as we will show in Section 3.2.5, we can essentially reduce the general case to this special case, and consequently obtain an algorithm for computing the optimal truthful mechanism whose runtime is asymptotically the same as Algorithm 3.1.

We first show (in Example 3.1) that, in general, there can be an arbitrarily large gap between the cost of the optimal deterministic mechanism and that of the optimal randomized mechanism,

**Algorithm 3.1:** Finding an optimal deterministic mechanism.

**Input:** The set of types $\Theta$, the principal's cost function $\{c_i\}_{i\in\Theta}$ for each type, the set of outcomes $\mathcal{O}$ (which encodes the agents' common utility function), and the reporting structure $R$.

**Output:** A deterministic truthful mechanism $M : \Theta \to \mathcal{O}$ minimizing the principal's cost.

Let $V \leftarrow (\Theta \times \mathcal{O}) \cup \{s, t\}$, $E \leftarrow \emptyset$;
Replace $R$ with its transitive closure (using the Floyd–Warshall algorithm):
**for** $i_2, i_1, i_3 \in \Theta$ *where* $(i_1, i_2) \in R$ *and* $(i_2, i_3) \in R$ **do**
    $R \leftarrow R \cup \{(i_1, i_3)\}$ ;
**end**
**for** *each type* $i \in \Theta$ **do**
    $E \leftarrow E \cup \{(s, (i, o_1), \infty)\}$ (add an edge from $s$ to $(i, o_1)$ with capacity $\infty$) ;
    **for** *each outcome* $j \in [m-1]$ **do**
        $E \leftarrow E \cup \{((i, o_j), (i, o_{j+1}), c_i(o_j))\}$ (add an edge from $(i, o_j)$ to $(i, o_{j+1})$ with capacity $c_i(o_j)$);
    **end**
    $E \leftarrow E \cup \{((i, o_m), t, c_i(o_m))\}$ (add an edge from $(i, o_m)$ to $t$ with capacity $c_i(o_m)$);
**end**
**for** *each pair of types* $(i_1, i_2)$ *where* $i_1 \neq i_2$ *and* $(i_1, i_2) \in R$*, and each outcome* $o_j \in \mathcal{O}$ **do**
    $E \leftarrow E \cup \{((i_2, o_j), (i_1, o_j), \infty)\}$ (add an edge from $(i_2, o_j)$ to $(i_1, o_j)$ with capacity $\infty$);
**end**
Compute an $s$-$t$ min-cut $(S, \overline{S})$ on graph $G = (V, E)$ ;
**for** *each type* $i \in \Theta$ **do**
    Let $M(i) = o_j$ where $j = \max\{j' \in [m] \mid (i, o_{j'}) \in S\}$;
**end**
**return** $M$;

even when restricted to truthful mechanisms and when all types share the same utility function.

**Example 3.1** (Gap between Deterministic and Randomized Mechanisms)**.** There are 2 types $\Theta = \{1, 2\}$ and 3 outcomes $\mathcal{O} = \{o_1 = 1, o_2 = 2, o_3 = 3\}$, which encode the common utility function. The principal's cost is given by $c_1(o_1) = c_1(o_3) = \infty$, $c_1(o_2) = 0$, $c_2(o_1) = c_2(o_3) = 0$, and $c_2(o_2) = \infty$. The reporting structure $R$ allows any type to report any other type, i.e., $R = \{(1, 1), (2, 2), (1, 2), (2, 1)\}$. Consider first the optimal truthful randomized mechanism, which as we argue below has cost 0. To make the principal's cost finite, the optimal truthful mechanism must assign outcome $o_2$ to type 1 with probability 1, which gives type 1 utility 2. To prevent misreporting, the mechanism must give type 2 the same expected utility. And again, to make the cost finite, it must never assign outcome $o_2$ to type 2. The unique way to satisfy the above is to assign to type 2 outcome $o_1$ with probability $1/2$, and $o_3$ with probability $1/2$.

Now consider any deterministic truthful mechanism. Any truthful mechanism must give both types the same utility to prevent misreporting. The only way to achieve this deterministically is to

assign the same outcome to both types. However, all 3 possibilities result in infinite total cost, so all deterministic truthful mechanisms have cost infinity.

Example 3.1 shows that Algorithm 3.1 in general does not find an (approximately) optimal truthful mechanism when randomized mechanisms are allowed. In such cases, one has to fall back to significantly slower algorithms, e.g., solving the straightforward LP formulation of the problem with $mn$ variables and $n^2$ constraints. It is worth noting that the LP formulation does not utilize the fact that types share an identical utility function. To address this issue, we identify an important special case where there does exist an optimal truthful mechanism that is deterministic: when the principal's cost is convex in the common utility function. More importantly, as we will show in Section 3.2.5, we can reduce the problem of finding the optimal randomized mechanism under general costs to the problem of finding the optimal mechanism with convex costs. First we formally define the notion of convex costs we use.

**Definition 3.1** (Convex Costs)**.** For any $i \in \Theta$, let the piecewise linear extension $c_i^\ell : [o_1, o_m] \to \mathbb{R}_+$ of $c_i$ be such that (1) for any $x \in \mathcal{O}$, $c_i^\ell(x) = c_i(x)$, and (2) for any $x \in [o_1, o_m] \setminus \mathcal{O}$,

$$c_i^\ell(x) = \frac{o_{j+1} - x}{o_{j+1} - o_j} \cdot c_i(o_j) + \frac{x - o_j}{o_{j+1} - o_j} c_i(o_{j+1}),$$

where $j = \max\{j' \in [m] \mid o_{j'} \leq x\}$. The principal's cost function $\{c_i\}_{i \in \Theta}$ is convex if for every $i \in \Theta$, the piecewise linear extension $c_i^\ell$ of $c_i$ is convex.

**Lemma 3.1** (Optimality of Deterministic Mechanisms with Convex Costs)**.** *When all types share the same utility function, and the principal's cost function is convex, there is an optimal truthful mechanism that is deterministic even with partial verification allowed.*

### 3.2.5 Reducing General Costs to Convex Costs

Lemma 3.1 together with Algorithm 3.1 provides an efficient way for finding optimal truthful mechanisms with convex costs (even when randomized mechanisms are allowed). One may still wonder if it is possible to design faster algorithms *in general* than solving the standard LP formulation, presumably by exploiting the additional structure that the agents share the same utility function. To this end, we observe that for computing optimal mechanisms, only the convex envelope of the principal's cost function matters. Given this observation, we show that finding optimal truthful mechanisms can be reduced very efficiently to finding optimal deterministic mechanisms.

We present Algorithm 3.2, which computes the optimal truthful mechanism and has the same asymptotic runtime as Algorithm 3.1. Algorithm 3.2 first computes the convex envelope of the principal's cost function, and then finds an optimal "deterministic" mechanism by calling Algorithm 3.1 with the same types and outcomes, but replacing the principal's cost function with its convex envelope. Algorithm 3.2 then recovers an optimal randomized mechanism from the "deterministic" one, by interpreting each "deterministic" outcome as a convex combination of outcomes in an optimal way. The following theorem establishes the correctness and time complexity of Algorithm 3.2.

**Theorem 3.4.** *Algorithm 3.2 finds an optimal (possibly randomized) truthful mechanism, in asymptotically the same time as Algorithm 3.1.*

**Algorithm 3.2:** Finding an optimal (possibly randomized) truthful mechanism.

**Input:** The set of types $\Theta$, the principal's cost function $\{c_i\}_{i\in\Theta}$ for each type, the set of outcomes $\mathcal{O}$ (which encodes the common utility function), and the reporting structure $R$.

**Output:** A truthful mechanism $M : \Theta \to \mathcal{O}$ minimizing the principal's cost.

**for** *each type* $i$ **do**

  Compute the convex envelope $c_i^- : [o_1, o_m] \to \mathbb{R}_+$ of $c_i$, defined such that for any $x \in [o_1, o_m]$,

$$c_i^-(x) = \min_{o\in\Delta(\mathcal{O}),\mathbb{E}[o]=x} \mathbb{E}[c_i(o)].$$

  Let $\widehat{c}_i$ be $c_i^-$ restricted to $\mathcal{O}$;

**end**

Run Algorithm 3.1 on input $(\Theta, \{\widehat{c}_i\}_{i\in\Theta}, \mathcal{O}, R)$. Let $\widehat{M}$ be the resulting deterministic mechanism;

**for** *each type* $i$ **do**

  $M(i) \leftarrow \operatorname{argmin}_{o\in\Delta(\mathcal{O}),\mathbb{E}[o]=\widehat{M}(i)} \mathbb{E}[c_i(o)]$ ;

**end**

**return** $M$;

Below we give a comparison between the time complexity of our algorithm, Algorithm 3.2, and that of the LP-based approach.[4] The current best algorithm for LP [23] takes time that translates to $\widetilde{O}(n^{2.37}m^{2.37} + n^{4.74})$[5] in our setting (this is, for example, at least $\widetilde{O}(n^{3.24}m^{1.5})$). The current best algorithm for $s$-$t$ min-cut [68] takes time that translates to $\widetilde{O}(n^{2.5}m^{1.5})$ in our setting. Moreover, in a typical classification setting, it is the number of outcomes (corresponding to "accept", etc.) $m$ that is small, and the number of types (e.g., "(CS major, highly competitive, female, international, ...)", "(math major, acceptable, male, domestic, ...)") $n$ is much larger. In such cases, the improvement becomes even more significant. Our results are theoretical, but practically, while there are highly optimized packages for LP, there are also highly optimized packages for max-flow / min-cut that are still much faster. Last but not least, in many practical settings, the principal has to implement a deterministic policy (it is hard to imagine college admissions explicitly made random), in which case our Algorithm 3.1 can be applied while LP generally does not give a solution.

## 3.3 Generalizing to Combinatorial Costs

In this section, we generalize the problem considered in the previous section, allowing the principal to have a combinatorial cost function over outcomes for each type. See Section 3.4 for a more detailed exposition.

[4] We note that a conclusive comparison is unrealistic since algorithms for both LP and min-cut keep being improved.

[5] $\widetilde{O}$ hides a poly-logarithmic factor.

**The combinatorial setting.** As before, let $\Theta = [n]$ be the set of types, $\mathcal{O} = \{o_j\}_{j\in[m]} \subseteq \mathbb{R}_+$ be the set of outcomes encoding the common utility function, and $R \subseteq \Theta \times \Theta$ be the reporting structure. The principal's cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ now maps a vector $O = (O^i)_i$ of outcomes for all types to the principal's cost $c(O)$. This subsumes the additive case, since one can set the cost function $c$ to be

$$c((O^i)_i) = \sum_{i\in\Theta} c_i(O^i).$$

Because the cost function is now combinatorial, it matters how the mechanism combines outcomes for different types. We therefore modify the definition of a randomized mechanism $M \in \Delta(\Theta \to \mathcal{O}) = \Delta(\mathcal{O}^\Theta)$, so that it allows correlation across different types. The principal's cost from using a truthful mechanism $M$ is then $c(M) = \mathbb{E}[c((M(i))_i)]$. For type $i$, the utility from executing mechanism $M$ is still $u_i(M) = \mathbb{E}[M(i)]$. $M$ is truthful iff for any $(i_1, i_2) \in R$, $u_{i_1}(M) \geq u_{i_2}(M)$. In the rest of the section, we present combinatorial generalizations of all our algorithmic and structural results given in the previous section.

**General vs. submodular cost functions.** Combinatorial functions in general are notoriously hard to optimize, even ignoring incentive issues. To see the difficulty, observe that a combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ over $\mathcal{O}^\Theta$ generally does not even admit a succinct representation (e.g., one whose size is polynomial in $m$ and $n$). It is therefore infeasible to take the entire cost function as input to an algorithm. To address this issue, the standard assumption in combinatorial optimization is that algorithms can access the combinatorial function through *value queries*. That is, we are given an oracle that can evaluate the combinatorial function $c$ at any point $O \in \mathcal{O}^\Theta$, obtaining the value $c(O)$ in constant time. For the rest of the chapter, we assume that our algorithm can access the cost function only through value queries.

Still, in order to minimize an *arbitrary* combinatorial function, in general one needs $\Omega(m^n)$ queries to obtain any nontrivial approximation. Despite that, there exist efficient algorithms for combinatorial minimization for an important subclass of cost functions, namely submodular functions.

**Definition 3.2** (Submodular Functions)**.** For any $O_1 = (O_1^i)_i \in \mathcal{O}^\Theta$ and $O_2 = (O_2^i)_i \in \mathcal{O}^\Theta$, let

$$O_1 \wedge O_2 = (\min(O_1^i, O_2^i))_i \text{ and } O_1 \vee O_2 = (\max(O_1^i, O_2^i))_i.$$

A combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ is submodular if for any $O_1, O_2 \in \mathcal{O}^\Theta$,

$$c(O_1) + c(O_2) \geq c(O_1 \wedge O_2) + c(O_1 \vee O_2).$$

In the rest of this section, we focus on submodular cost functions. For this important special case, we give efficient algorithms for finding optimal truthful deterministic / randomized mechanisms, as well as a sufficient condition for the existence of an optimal mechanism that is deterministic.

**Finding optimal deterministic mechanisms.** First we present a polynomial-time combinatorial algorithm for finding optimal truthful deterministic mechanisms with partial verification, when the cost function is submodular.

**Theorem 3.5.** *There exists a polynomial-time algorithm which accesses the cost function via value queries only, and computes an optimal deterministic truthful mechanism when partial verification is allowed and the cost function is submodular.*

**Sufficient condition for the optimality of deterministic mechanisms.** Restricted to additive cost functions, Lemma 3.1 gives a sufficient condition under which there exists an optimal mechanism that is deterministic. We present below a combinatorial version of this structural result when the outcome space is binary, i.e., when $m = 2$.

**Theorem 3.6** (Optimality of Deterministic Mechanisms with Binary Outcomes)**.** *When the outcome space is binary, i.e., $|\mathcal{O}| = 2$, and the principal's cost function is submodular, there is an optimal truthful mechanism that is deterministic, even when partial verification is allowed.*

**Computing optimal randomized mechanisms.** Finally we give an algorithm for finding an optimal mechanism with arbitrary submodular cost functions.

**Theorem 3.7.** *When the cost function $c$ is submodular and bounded, for any desired additive error $\varepsilon > 0$, there is an algorithm which finds an $\varepsilon$-approximately optimal (possibly randomized) truthful mechanism*[6] *in time $\mathrm{poly}(n, m, \log(1/\varepsilon))$, even if partial verification is allowed.*

## 3.4 Generalizing to Combinatorial Costs

In this section, we generalize the problem considered in the previous section, allowing the principal to have a combinatorial cost function over outcomes for each type. The problem studied in the previous section can be viewed as a special case (where the principal's cost function is additive over types) of this general problem. Before we proceed to the formal definition of the problem, to better motivate combinatorial cost functions, consider the following example.

**Example 3.2.** Suppose in addition to an additive cost function $\{c_i\}_{i\in\Theta}$, the principal has to pay an overhead cost $c_0 > 0$ as long as any type receives a nontrivial outcome, i.e., if there exists $i \in \Theta$, such that $M(i) \in \mathcal{O} \setminus \{o_1\}$. In such cases, the principal's overall cost from executing a truthful deterministic mechanism $M$ can be written as

$$c(M) = \sum_{i\in\Theta} c_i(M(i)) + c_0 \cdot \mathbb{I}[\exists i \in \Theta : M(i) \neq o_1],$$

where $\mathbb{I}[\cdot]$ is the indicator of a statement.

In the above rather natural example, the principal's cost is no longer additive over types. As a result, there is no way to properly formulate Example 3.2 using our previous definitions. We generalize the principal's cost function, as well as the definition of mechanisms, as follows.

[6] An $\varepsilon$-approximately optimal truthful mechanism is a truthful mechanism whose expected cost is at most $\varepsilon$ larger than the minimum possible cost of any truthful mechanism.

**Notation.** As before, let $\Theta = [n]$ be the set of types, $\mathcal{O} = \{o_j\}_{j\in[m]} \subseteq \mathbb{R}_+$ be the set of outcomes encoding the common utility function, and $R \subseteq \Theta \times \Theta$ be the reporting structure. The principal's cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ now maps a vector $O = (O^i)_i$ of outcomes for all types to the principal's cost $c(O)$. This subsumes the additive case, since one can set the cost function $c$ to be

$$c((O^i)_i) = \sum_{i\in\Theta} c_i(O^i).$$

Because the cost function is now combinatorial, it matters how the mechanism combines outcomes for different types. We therefore modify the definition of a (possibly randomized) mechanism $M \in \Delta(\Theta \to \mathcal{O}) = \Delta(\mathcal{O}^\Theta)$, such that it allows correlation across different types. The principal's cost from executing a truthful mechanism $M$ is then

$$c(M) = \mathbb{E}[c((M(i))_i)].$$

We treat $M$ as a distribution or a random variable over $\mathcal{O}^\Theta$ interchangeably. Note that each type's utility is still independent of what other types get. So for type $i$, the utility from executing mechanism $M$ is still

$$u_i(M) = \mathbb{E}[M(i)].$$

And $M$ is truthful iff for any $(i_1, i_2) \in R$,

$$u_{i_1}(M) \geq u_{i_2}(M).$$

In the rest of the section, we present combinatorial generalizations of all our algorithmic and structural results given in the previous section.

### 3.4.1 General vs. Submodular Cost Functions

Combinatorial functions in general are notoriously hard to optimize, even ignoring incentive issues. To see the difficulty, observe that a combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ over $\mathcal{O}^\Theta$ generally does not even admit a succinct representation (e.g., one whose size is polynomial in $m$ and $n$). It is therefore infeasible to take the entire cost function as input to an algorithm.

To address this issue, the standard assumption in combinatorial optimization is that algorithms can access the combinatorial function through *value queries*. That is, we are given an oracle that can evaluate the combinatorial function $c$ at any point $O \in \mathcal{O}^\Theta$, obtaining the value $c(O)$ in constant time. For the rest of the chapter, we assume that our algorithm can only access the cost function only through value queries.

Now suppose we are to design an algorithm to minimize an arbitrary combinatorial cost function, without any additional constraint. That is, given a combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$, we wish to find a point $O \in \mathcal{O}^\Theta$, such that $c(O)$ is minimized over $\mathcal{O}^\Theta$. The example below shows that any algorithm which interacts with $c$ only through value queries needs $\Omega(m^n)$ queries to obtain any nontrivial approximation to the above seemingly basic problem.

**Example 3.3.** Let the cost function $c$ be generated in the following random way. A point $O^*$ is drawn from $\mathcal{O}^\Theta$ uniformly at random. $c$ is then constructed such that $c(O^*) = 0$, and $c(O) = 1$ for any $O \neq O^*$. To minimize $c$, the algorithm has to find $O^*$. This is equivalent to guessing a uniformly random number among $m^n$ numbers. To guess $O^*$ successfully with constant probability, one has to make $\Omega(m^n)$ guesses.

The above issue has been identified in combinatorial optimization since decades ago. Despite the fact that general combinatorial cost functions are hard to minimize, researchers have developed efficient algorithms for combinatorial minimization for an important subclass of cost functions, namely submodular functions, defined below.

**Definition 3.3** (Submodular Functions)**.** For any $O_1 = (O_1^i)_i \in \mathcal{O}^\Theta$ and $O_2 = (O_2^i)_i \in \mathcal{O}^\Theta$, let

$$O_1 \wedge O_2 = (\min(O_1^i, O_2^i))_i \quad \text{and} \quad O_1 \vee O_2 = (\max(O_1^i, O_2^i))_i.$$

A combinatorial cost function $c : \mathcal{O}^\Theta \to \mathbb{R}_+$ is submodular, if for any $O_1, O_2 \in \mathcal{O}^\Theta$,

$$c(O_1) + c(O_2) \geq c(O_1 \wedge O_2) + c(O_1 \vee O_2).$$

In the rest of the section, we focus on submodular cost functions. For this important special case, we give efficient algorithms for finding optimal truthful deterministic/randomized mechanisms, as well as a sufficient condition for the existence of an optimal mechanism that is deterministic.

### 3.4.2 Finding Optimal Deterministic Mechanisms

First we present a polynomial-time combinatorial algorithm for finding optimal truthful deterministic mechanisms with partial verification, when the cost function is submodular. The algorithm is based on the key observation that the space of truthful deterministic mechanisms is a distributive lattice (defined below in Lemma 3.2). Given this observation, it is known that the problem can be reduced to minimizing a submodular function without additional constraints, which can be solved efficiently.

**Lemma 3.2.** *Fix the set of types $\Theta$, the set of outcomes $\mathcal{O}$, and the reporting structure $R$. Let $\mathcal{T} \subseteq \mathcal{O}^\Theta$ be the space of all possible ways of assigning outcomes to types, such that no type has the incentive to misreport. That is,*

$$\mathcal{T} = \{O = (O^i)_i \in \mathcal{O}^\Theta \mid \forall (i_1, i_2) \in R,\, O^{i_1} \geq O^{i_2}\}.$$

*Then $\mathcal{T}$ is a distributive lattice, i.e., $\mathcal{T}$ satisfies the following conditions.*

- *For any $O_1, O_2 \in \mathcal{T}$, $O_1 \wedge O_2 \in \mathcal{T}$, and $O_1 \vee O_2 \in \mathcal{T}$.*
- *For any $O_1, O_2, O_3 \in \mathcal{T}$, $O_1 \vee (O_2 \wedge O_3) = (O_1 \vee O_2) \wedge (O_1 \vee O_3)$.*

*Proof.* Consider the first property. Fix any $O_1, O_2 \in \mathcal{T}$, and let $O_- = O_1 \wedge O_2$, and $O_+ = O_1 \vee O_2$. For any $(i_1, i_2) \in R$, since $O_1^{i_1} \geq O_1^{i_2}$ and $O_2^{i_1} \geq O_2^{i_2}$, we have

$$O_-^{i_1} = \min(O_1^{i_1}, O_2^{i_1}) \geq \min(O_1^{i_2}, O_2^{i_2}) = O_-^{i_2}.$$

This implies $O_- \in \mathcal{T}$. Similarly we may show $O_+ \in \mathcal{T}$. In other words, the first property holds.

For the second property, simply consider the $i$-th coordinate for any $i \in \Theta$. Fix $O_1, O_2, O_3 \in \mathcal{T}$, we have

$$\max(O_1^i, \min(O_2^i, O_3^i)) = \min(\max(O_1^i, O_2^i), \max(O_1^i, O_3^i)).$$

Since this is true for any $i$, the second property follows immediately. □

Given Lemma 3.2, we can apply the algorithm and the reduction by Schrijver [92] to obtain an efficient algorithm directly.

**Corollary 3.1.** *There exists a polynomial-time algorithm which accesses the cost function via value queries only, and computes an optimal deterministic truthful mechanism when partial verification is allowed and the cost function is submodular.*

*Proof.* Let $\mathcal{T}$ be the family of truthful assignments defined in Lemma 3.2. The problem of finding an optimal deterministic truthful mechanism can be equivalently formulated as the following optimization problem.

$$\min_{O \in \mathcal{T}} c(O).$$

This can be solved by applying the reduction in Section 6 of [92][7] to any algorithm for minimizing submodular functions (e.g., the one given in [92]). □

We remark that the algorithm by Schrijver [92] can be applied as well in the classical additive setting, but due to its generality, is significantly less efficient than Algorithm 3.1.

### 3.4.3 Sufficient Condition for the Optimality of Deterministic Mechanisms

We have shown in Example 3.1 that the gap between deterministic and randomized mechanisms can be arbitrarily large. Restricted to additive cost functions, Lemma 3.1 gives a sufficient condition under which there exists an optimal mechanism that is deterministic. We present in this subsection a combinatorial version of this structural result when the outcome space is binary, i.e., when $m = 2$.

**Lemma 3.3** (Optimality of Deterministic Mechanisms with Binary Outcomes)**.** *When the outcome space is binary, i.e., $|\mathcal{O}| = 2$, and the principal's cost function is submodular, there is an optimal truthful mechanism that is deterministic, even when partial verification is allowed.*

*Proof.* The overall plan is similar to that of the proof of Lemma 3.1. We begin with a (possibly randomized) optimal truthful mechanism $M$, and show that without loss of generality, we may assume its support has some monotone structure. We then round this mechanism, such that the resulting deterministic mechanism is always truthful, and the expected cost of the rounded mechanism is equal to the cost of $M$.

Without loss of generality, suppose $\mathcal{O} = \{0, 1\}$. Observe that $\mathcal{O}^\Theta$ is isomorphic to $2^\Theta$, so in the rest of the proof, we interchangeably represent an outcome vector $O$ as a subset of $\Theta$, i.e., the set

$$\{i \in \Theta \mid O^i = 1\}.$$

[7] Although the reduction presented therein is for ring families, one may check it also works for distributive lattices.

Let $M$ be any (possibly randomized) optimal truthful mechanism. Below we treat $M$ as a random variable distributed over $\mathcal{O}^\Theta$, or interchangeably, a random subset of $\Theta$. For any $O \in \mathcal{O}^\Theta$, let $p(O) = \Pr[M = O]$ be the probability that $M$ assigns outcomes $O$ to types. We further require $M$ to maximize the following potential function among all optimal truthful mechanisms.

$$\mathbb{E}[|M|^2] = \sum_{O \in \mathcal{O}^\Theta} p(O) \cdot |O|^2.$$

We argue below that for such an $M$, no two outcome vectors in the support of $M$ "cross."

We say two outcome vectors $O_1$ and $O_2$ (represented as sets) cross, if $O_1 \not\subseteq O_2$ and $O_2 \not\subseteq O_1$. Toward a contradiction, suppose $O_1$ and $O_2$ cross, where without loss of generality $p(O_1) \geq p(O_2) > 0$. Let $p = p(O_2)$. We show that moving probability mass from $O_1$ and $O_2$ to $O_1 \cap O_2$ and $O_1 \cup O_2$ simultaneously preserves truthfulness, does not increase the cost, and strictly increases the potential of $M$, which contradicts the choice of $M$.

To be precise, we decrease $p(O_1)$ and $p(O_2)$ simultaneously by $p$, and increase $p(O_1 \cap O_2)$ and $p(O_1 \cup O_2)$ simultaneously by $p$. To see why truthfulness is preserved, observe that the expected utility of any type does not change after the modification. The change of the cost can be written as

$$\begin{aligned}
\Delta c(M) &= p \cdot (c(O_1 \cap O_2) + c(O_1 \cup O_2) - c(O_1) - c(O_2)) \\
&\leq p \cdot (c(O_1) + c(O_2) - c(O_1) - c(O_2)) && \text{(submodularity of } c\text{)} \\
&= 0,
\end{aligned}$$

so the cost does not increase. Finally, the change of the potential is

$$p \cdot (|O_1 \cap O_2|^2 + |O_1 \cup O_2|^2 - |O_1|^2 - |O_2|^2).$$

It is easy to check the above is strictly positive as long as $O_1$ and $O_2$ cross.

From now on we assume no two outcomes in the support of $M$ cross, or equivalently, the support of $M$ is a family of nested subsets of $\Theta$. For any $r \in [0, 1]$, let $M_r \subseteq \Theta$ be such that

$$M_r = \{i \in \Theta \mid u_i \geq r\}.$$

Observe that $M_r$ is truthful for any $r \in [0, 1]$. In fact, for any $(i_1, i_2) \in R$, we always have

$$i_2 \in M_r \Longrightarrow i_1 \in R.$$

Truthfulness then follows.

Consider the random (but not randomized) mechanism $M_r$ when $r$ is uniformly distributed over $[0, 1]$. We argue below that the expected cost of $M_r$ is precisely that of $M$. In fact, $M_r$ and $M$ are even identically distributed. For any $i \in \Theta$, let the expected utility of type $i$ be $u_i = \Pr[i \in M]$. Without loss of generality, suppose $M$ satisfies $u_i \geq u_{i+1}$ for any $i \in [n - 1]$. To show $M_r$ and $M$ are identically distributed, we only need to show that

$$p(O) > 0 \Longrightarrow \exists i \in \Theta, O = [i].$$

In other words, $M$ only assigns outcomes that are prefixes of $\Theta$. Given this, $M_r$ is the only distribution that gives each type $i$ expected utility $u_i$ simultaneously.

To see why the above claim is true, suppose there exists some $O \subseteq \Theta$ where $1 \notin O$ and $p(O) > 0$. Then since the support of $M$ is a nested family of subsets of $\Theta$, for any $i \in O$, regardless of the realization of $M$, we always have

$$1 \in M \implies i \in O.$$

As a result,

$$u_i = \sum_{O' \subseteq \Theta : i \in O'} p(O') \geq p(O) + \sum_{O' \subseteq \Theta : 1 \in O'} p(O') = p(O) + u_1 > u_1,$$

a contradiction. This concludes the proof. □

We make the following remarks regarding Lemma 3.3.

- The binary outcomes assumption, despite being more restrictive than the general model, still captures many real-life applications. In particular, it models binary classification problems where one label is more desirable than the other for all agents. Common examples include hiring decisions, university admissions, etc. Moreover, such decisions are generally correlated over types (e.g., universities cannot admit too many students) — this is captured by the principal's submodular cost function.
- The proof of Lemma 3.3 can be alternatively interpreted in the following way. Without loss of generality, any optimal mechanism corresponds to a point on the convex envelope of the principal's cost function. And for submodular cost functions particularly, this convex envelope happens to coincide with the Lovász extension (see [52]), which can be derandomized into deterministic mechanisms preserving truthfulness. We will further develop this intuition in the next result, which is an efficient algorithm for finding optimal truthful mechanisms for any submodular function.

### 3.4.4 An Efficient Algorithm for Computing Optimal Randomized Mechanisms

In this subsection, we present an algorithm for finding an optimal mechanism with arbitrary submodular cost functions.

Our algorithm, Algorithm 3.3, again builds on the intuition that for optimal mechanisms, only the convex envelope of the cost function matters. The problem of finding optimal mechanisms can therefore be formulated as a convex program. However, unlike in Algorithm 3.2, with submodular cost functions, it is not clear how one can efficiently evaluate the convex envelope of the cost function.

To get around this issue, instead of parametrizing by the target utilities, we parametrize the convex envelope by the marginal probabilities $\{p_{i,j}\}_{i \in \Theta, j \in [m]}$, where $p_{i,j}$ is the probability that type $i$ gets outcome $o_j$. One of the key ingredients of Algorithm 3.3 is a subroutine (Algorithm 3.4) which efficiently interprets each point on the convex envelope as a convex combination of integral

**Algorithm 3.3:** Finding an optimal mechanism with submodular cost functions.

**Input:** The set of types $\Theta$, the principal's submodular cost function $c$, the set of outcomes $O$ (which encodes the common utility function), the reporting structure $R$, and a precision parameter $\varepsilon$.

**Output:** An optimal truthful mechanism $M \in \Delta(\Theta \to \mathcal{O})$.

Compute an $\varepsilon$-approximately optimal solution $p^*_{i,j}$ to the following convex program using the ellipsoid method (see, e.g., [15]), and call Algorithm 3.4 with parameters $\{p_{i,j}\}$ for evaluating $\mathrm{prob}(\cdot \mid \{p_{i,j}\})$

$$\begin{aligned}
\min \quad & \sum_{O \in \mathcal{O}^\Theta} \mathrm{prob}(O \mid \{p_{i,j}\}_{i\in\Theta, j\in[m]}) \cdot c(O) \\
\text{s.t.} \quad & \sum_{j\in[m]} p_{i_1,j} \cdot o_j \geq \sum_{j\in[m]} p_{i_2,j} \cdot o_j && \forall (i_1, i_2) \in R \\
& \sum_{j\in[m]} p_{i,j} = 1 && \forall i \in \Theta \\
& p_{i,j} \geq 0 && \forall i \in \Theta, j \in [m];
\end{aligned}$$

**for** *each* $O \in \mathcal{O}^\Theta$ **do**
    $\Pr[M = O] \leftarrow \mathrm{prob}(O \mid \{p^*_{i,j}\})$;
**end**
**return** $M$;

points, corresponding to a distribution over combinations of outcomes. In other words, given the desired marginal probabilities, Algorithm 3.4 finds a randomized truthful mechanism realizing these marginal probabilities which minimizes the principal's expected cost.

**Theorem 3.8.** *When the cost function $c$ is submodular and bounded, for any desired additive error $\varepsilon > 0$, Algorithm 3.3 finds an $\varepsilon$-approximately optimal (possibly randomized) truthful mechanism* [8] *in time* $\mathrm{poly}(n, m, \log(1/\varepsilon))$*, even if partial verification is allowed.*

*Proof.* Consider the program in Algorithm 3.3. Observe there are $mn$ variables, namely $\{p_{i,j}\}$, and $O(n^2 + mn)$ linear constraints in the program. In order for the program to be efficiently solvable, we only need to show the following three claims hold.

- The subroutine for evaluating the convex envelope, Algorithm 3.4, runs in polynomial time.
- The objective is convex in $\{p_{i,j}\}$.
- A first-order oracle, which computes a (sub)gradient of the objective function at any point, can be efficiently implemented. (The ellipsoid method also requires an efficient separation oracle, which for our program exists straightforwardly, since there are only $\mathrm{poly}(n, m)$ linear constraints.)

[8] An $\varepsilon$-approximately optimal truthful mechanism is a truthful mechanism whose expected cost is at most $\varepsilon$ larger than the minimum possible cost of any truthful mechanism.

**Algorithm 3.4:** Algorithm for interpreting the convex envelope.

**Input:** Marginal probabilities $\{p_{i,j}\}_{i,j}$, where $p_{i,j}$ is the desired probability that type $i$ gets outcome $o_j$.

**Output:** A distribution $\{\mathrm{prob}(O \mid \{p_{i,j}\})\}_{O \in \mathcal{O}^\Theta}$ over combinations of outcomes.

1 Let $\mathrm{prob}(O) \leftarrow 0$ for all $O \in \mathcal{O}^\Theta$;
2 **while** *there is some* $p_{i,j} > 0$ **do**
3  **for** *each* $i \in \Theta$ **do**
4  $t_i \leftarrow \max\{j \in [m] \mid p_{i,j} > 0\}$;
5  **end**
6  $\delta \leftarrow \min_i p_{i,t_i}$; $\quad \mathrm{prob}((t_i)_{i\in[n]}) \leftarrow \delta$;
7  **for** *each* $i \in \Theta$ **do**
8  $p_{i,t_i} \leftarrow p_{i,t_i} - \delta$;
9  **end**
10  return $\{\mathrm{prob}(O)\}_{O \in \mathcal{O}^\Theta}$;
11 **end**

The second claim and the third claim also imply the correctness of the Algorithm. Below we prove the three claims.

Consider the first claim. We only need to show that the while-loop at line 2 repeats only polynomially many times. Consider the following potential function $\phi$.

$$\phi(\{p_{i,j}\}) = \sum_{i \in \Theta} \max\{j \in [m] \mid p_{i,j} > 0\}.$$

Before line 2, the value of $\phi$ is at most $mn$. Observe that $\phi$ is monotone in $\{p_{i,j}\}$, and the latter never increase during the execution of the loop. Moreover, in each repetition of the loop, $\phi$ decreases at least by 1. This is because after the update in line 9, for some $i \in \Theta$, $p_{i,t_i}$ becomes 0, and as a result, $\max\{j \in [m] \mid p_{i,j} > 0\}$ decreases at least by 1. When $\phi$ becomes 0, it must be the case that $p_{i,j} = 0$ for any $i \in \Theta, j \in [m]$, so the loop terminates. Therefore the while-loop repeats at most $mn$ times, which implies the first claim.

Now consider the second claim. We show that in Algorithm 3.4, the output distribution $\{p(O)\}$ minimizes the expected cost

$$\sum_{O \in \mathcal{O}^\Theta} p(O) \cdot c(O)$$

among all distributions whose marginals are $\{p_{i,j}\}$ — this is equivalent to the second claim. We first prove the following characterization of the output distribution $\{p(O)\}$.

**Lemma 3.4.** *The output distribution $\{p(O)\}$ of Algorithm 3.4 is the only distribution over $O^\Theta$ satisfying the following properties.*

- *$\{p(O)\}$ induce the input marginal probabilities $\{p_{i,j}\}$ over type-outcome pairs.*
- *For any $O_1, O_2 \in \mathcal{O}^\Theta$, if $O_1 \wedge O_2 \notin \{O_1, O_2\}$, then either $p(O_1) = 0$ or $p(O_2) = 0$. In other words, no two combinations of outcomes in the support of $\{p(O)\}$ cross.*

*Proof.* The first bullet point is clear from the construction of $\{p(O)\}$. We therefore focus on the second bullet point. We first show that for any marginal probabilities $\{p_{i,j}\}$ and distribution $\{p'(O)\}$, if (1) $\{p'(O)\}$ induce $\{p_{i,j}\}$ and no two combinations of outcomes in the support of $\{p'(O)\}$ cross, then the topmost combination of outcomes $O_t$, where

$$O_t^i = \max\{o_j \mid j \in [m], p_{i,j} > 0\},$$

must have probability exactly

$$p'(O_t) = \min_{i \in \Theta} p_{i,t_i},$$

where $t_i = \max\{j \in [m] \mid p_{i,j} > 0\}$.

Let $\delta = \min_{i \in \Theta} p_{i,t_i}$. Observe that $p'(O_t) \leq \delta$. Suppose toward a contradiction that $p'(O_t) < \delta$. Since no two combinations of outcomes in the support of $\{p'(O)\}$ cross, we can order the support of $\{p'(O)\}$ as $O_1, \ldots, O_\ell$, where $\ell$ is the size of the support, such that for any $i \in [\ell - 1]$,

$$O_i \wedge O_{i+1} = O_{i+1}.$$

Clearly we have $O_t \wedge O_1 = O_1$. Consider the following two cases.

- $O_t \neq O_1$. In other words, there is some $i^* \in \Theta$, such that $O_t^{i^*} > O_1^{i^*}$. As a result, for any $k \in [\ell]$,

  $$O_k^{i^*} \leq O_1^{i^*} < O_t^{i^*} \leq o_{t_{i^*}}.$$

  Then we have

  $$0 < \delta \leq p_{i^*,t_{i^*}} = \sum_{k \in [\ell]} p'(O_k) \cdot \mathbb{I}[O_k^{i^*} = o_{t_{i^*}}] = 0,$$

  a contradiction.
- $O_t = O_1$. There is some $i^* \in \Theta$, such that $O_1^{i^*} > O_2^{i^*}$. As a result, for any $2 \leq k \leq \ell$,

  $$O_k^{i^*} \leq O_2^{i^*} < O_1^{i^*} = o_{t_{i^*}}.$$

  So we have

  $$\delta \leq p_{i^*,t_{i^*}} = \sum_{k \in [\ell]} p'(O_k) \cdot \mathbb{I}[O_k^{i^*} = o_{t_{i^*}}] \leq p'(O_1) < \delta,$$

  a contradiction.

So in any case, we must have $p'(O_t) = \delta$.

Now observe that the above argument does not depend on the fact that for any $i \in \Theta$, $\sum_{j \in [m]} p_{i,j} = 1$. Therefore, we can repeatedly apply the characterization of the probability of the topmost combination. That is, we first compute the probability of the topmost combination, and subtract the marginal probabilities contributed by this topmost combination from $\{p_{i,j}\}$. For the new marginal probabilities, the characterization still applies to the new topmost combination, by which we can determine the probability of that combination. This is precisely the procedure implemented in Algorithm 3.4. By repeatedly applying the characterization, we obtain the unique distribution over $\mathcal{O}^\Theta$ satisfying the conditions of the lemma, which is the output distribution $\{p(O)\}$ of Algorithm 3.4. □

Given Lemma 3.4, we then consider any distribution $\{p'(O)\}$ which (1) induces marginal probabilities $\{p_{i,j}\}$, (2) minimizes the expected cost, and (3) among all distributions satisfying (1) and (2), maximizes the potential function $w$, defined as

$$w(p') = \sum_{O \in \mathcal{O}^\Theta} p'(O) \cdot \left( \sum_{i \in \Theta} O^i \right)^2 .$$

The goal is to show such a distribution $\{p'(O)\}$ satisfies the conditions of Lemma 3.4, and therefore coincides with $\{p(O)\}$, the output of Algorithm 3.4. Moreover, such a distribution has the additional property, that it minimizes the expected cost. In other words, the output distribution of Algorithm 3.4 minimizes the expected cost, which is equivalent to the second claim at the beginning of the proof.

To achieve the above goal, we only need to show that the distribution $\{p'(O)\}$ chosen above has the property, that no two combinations of outcomes in the support of $\{p'(O)\}$ cross. Suppose otherwise, i.e., there exist $O_1, O_2 \in \mathcal{O}^\Theta$, such that $O_1 \wedge O_2 \notin \{O_1, O_2\}$ and $p'(O_1) > p'(O_2) > 0$. Let $q = p'(O_2)$. We show that subtracting $q$ from $p'(O_1)$ and $p'(O_2)$ simultaneously and adding $q$ to $p'(O_1 \wedge O_2)$ and $p'(O_1 \vee O_2)$ simultaneously (1) preserves the marginal probabilities, (2) does not increase the expected cost of $\{p'(O)\}$, and (3) strictly increases the potential function $w$, thus leading to a contradiction. (1) clearly holds. (2) follows from the submodularity of $c$, i.e.,

$$q \cdot (c(O_1) + c(O_2)) \geq q \cdot (c(O_1 \wedge O_2) + c(O_1 \vee O_2)).$$

And finally, (3) follows from elementary calculation, i.e., whenever $O_1$ and $O_2$ cross,

$$\left( \sum_{i \in \Theta} O_1^i \right)^2 + \left( \sum_{i \in \Theta} O_2^i \right)^2 < \left( \sum_{i \in \Theta} \min(O_1^i, O_2^i) \right)^2 + \left( \sum_{i \in \Theta} \max(O_1^i, O_2^i) \right)^2 .$$

This establishes the second claim.

As for the third claim, i.e., the existence of an efficient first-order oracle, observe that the distribution $\{\mathrm{prob}(O \mid \{p_{i,j}\})\}$ output by Algorithm 3.4 is piecewise linear in $\{p_{i,j}\}$. On the other hand, the objective function is linear in the output distribution, and is therefore piecewise linear in $\{p_{i,j}\}$. This implies that (sub)gradients of the objective function can be easily computed, and concludes the proof of the theorem. □

We make a few remarks regarding Algorithm 3.3.

- In addition to the ellipsoid method, one may also apply gradient-based methods, e.g., projected gradient descent, to solve the convex program in Algorithm 3.3. Gradient-based methods generally perform better in practice, and they usually have better dependence on $m$ and $n$ but worse (polynomial) dependence on $1/\varepsilon$.
- Observe that Algorithm 3.3 outputs a randomized mechanism such that, in its support, no two combinations of outcomes cross. Therefore, when restricted to binary outcomes ($m = 2$), Lemma 3.3 gives a way to round the randomized mechanism output by Algorithm 3.3 into a deterministic one. This gives an alternative way (in addition to Corollary 3.1) of computing optimal deterministic mechanisms restricted to binary outcomes.

## 3.5 Omitted Definitions and Remarks in Section 3.2

### 3.5.1 Single-Peaked Preferences

Below we give a definition of single-peaked preferences.

**Definition 3.4.** Let $O$ be the space of outcomes, $\Theta$ the space of types, and for each $i \in \Theta$, $u_i : O \to \mathbb{R}_+$ the utility function of an agent of type $i$. $\{u_i\}_i$ are single-peaked if there exists an ordering $\prec$ over $O$, such that for each type $i$ the following holds: there exists a most preferred outcome $o^i$. Moreover, for any two outcomes $o_{j_1} \prec o_{j_2}$,

- if $o_{j_2} \prec o^i$, then $u_i(o_{j_1}) \leq u_i(o_{j_2}) \leq u_i(o^i)$;
- if $o^* \prec o_{j_1}$, then $u_i(o^i) \geq u_i(o_{j_1}) \geq u_i(o_{j_2})$.

In words, the above definition says that the outcomes can be ordered in a line, such that for each type, there exists a most preferred outcome. Moreover, on both sides of this most preferred outcome, the closer an outcome is to the most preferred the outcome, the higher the utility is for that outcome.

### 3.5.2 Remarks on Algorithm 3.1

We make two remarks regarding Algorithm 3.1.

- For finding an optimal deterministic mechanism, the precise values of the agents' utility functions do not matter. Consequently, Algorithm 3.1 works as long as all types *order* the outcomes in the same way.
- With minor modifications, Algorithm 3.1 can handle *costly* misreporting, in which there is a fixed (non-negative) cost for type $i$ to report as type $i'$. Partial verification is a special case of costly misreporting: reporting either costs the agent $0$ or $\infty$, and the reporting structure $R$ is the set of all reporting actions which cost $0$. The key modification which allows Algorithm 3.1 to handle costly misreporting is that the edges used to model the reporting structure can be diagonal (as opposed to horizontal), where the slope of the edge depends on each type's utility function and the cost of misreporting. We will not expand on this in the current chapter.

### 3.5.3 Remarks on Lemma 3.1

We make a few remarks regarding Lemma 3.1.

- The proof we present is a combination of several concrete arguments. There is an alternative relatively high-level, and sometimes more useful, interpretation of the lemma, which is based on a convex program formulation of the problem. We will make heavy use of this alternative interpretation in the rest of the chapter, especially when dealing with randomized mechanisms.
- Throughout the chapter we assume payments are not allowed. One may show that with payments, there always exists an optimal truthful mechanism that is deterministic, as long as both agents and the principal value payments linearly. Moreover, there exist relatively

simple algorithms for computing an optimal mechanism with payments. We will not expand on this in the current chapter.

### 3.5.4 Remarks on Algorithm 3.2

We make a few remarks regarding Algorithm 3.2.

- Algorithm 3.2 gives a constructive proof that finding an optimal truthful mechanism is always no harder than finding an optimal truthful deterministic mechanism with convex costs. As a result, a faster algorithm for the latter problem would imply a faster algorithm for the former.
- As a byproduct, Algorithm 3.2 shows that in general, to achieve the minimum cost, it suffices to randomize only between two outcomes for each type, .

## 3.6 Omitted Proofs in Section 3.2

*Proof of Theorem 3.1.* We give a reduction from MinSAT. Fix a MinSAT instance with $n$ variables, $\{x_i\}_{i\in[n]}$, and $m$ clauses, $\{C_j\}_{j\in[m]}$, and let $\ell_{j,k} \in C_j$ be the $k$-th literal in clause $C_j$. We construct an AMD instance as follows.

- Create a type for each variable, each literal, and each clause, i.e., $\Theta = \{x_i, x_i^+, x_i^-\}_{i\in[n]} \cup \{C_j\}_{j\in[m]}$.
- There are two possible outcomes, $\mathcal{O} = \{o^+, o^-\}$. Moreover, for any type $\theta \in \Theta$, $u_\theta(o^+) = 1$ and $u_\theta(o^-) = 0$.
- The principal's cost is as follows.
  - For each literal $\ell$, $c_\ell(o^+) = c_\ell(o^-) = 0$.
  - For each variable $x_i$, $c_{x_i}(o^+) = 0$ and $c_{x_i}(o^-) = m + 1$, so any optimal mechanism never assigns $o^-$ to a variable.
  - For each clause $C_j$, $c_{C_j}(o^+) = 1$ and $c_{C_j}(o^-) = 0$, so any optimal mechanism minimizes the number of clauses which get outcome $o^+$.
- The reporting structure $R$ is as follows.
  - Each literal $\ell$ can only report itself.
  - Each variable $x_i$ can report itself and its two literals $x_i^+$ and $x_i^-$.
  - Each clause $C_j$ can report itself, all variables, and all literals $\ell_{j,k} \in C_j$ contained in $C_j$.

Now consider the structure of optimal solutions for the above AMD instance. First observe that without loss of generality, any optimal solution assigns $o^+$ only to types which report literals. Moreover, for each variable $x_i$, any optimal solution assigns $o^+$ to exactly one of $x_i^+$ and $x_i^-$. So the problem boils down to choosing between the two literals for each variable.

On the other hand, each clause $C_j$ will report any literal that is contained in $C_j$ and assigned outcome $o^+$, as long as possible. Whenever this happens, the principal incurs cost 1 from this clause. In other words, the principal incurs cost 1 from a clause iff one of the literals contained in

the clause is assigned outcome $o^+$, i.e., iff the clause is satisfied. The total cost of the mechanism is $k$, where $k$ is the number of clauses satisfied. This encodes precisely the MinSAT instance. □

*Proof of Theorem 3.2.* Consider the following reduction from MinSAT. Fix a MinSAT instance with $n$ variables, $\{x_i\}_{i\in[n]}$, and $m$ clauses, $\{C_j\}_{j\in[m]}$, and let $\ell_{j,k} \in C_j$ be the $k$-th literal in clause $C_j$. We construct an AMD instance as follows.

- Create a type for each variable, each literal, and each clause, i.e., $\Theta = \{x_i, x_i^+, x_i^-\}_{i\in[n]} \cup \{C_j\}_{j\in[m]}$.
- There are three possible outcomes, $\mathcal{O} = \{o^+, o^-, o^0\}$.
- Let $N_1 >> N_2 > m$ be large (but polynomial in $n$ and $m$) numbers. The principal's cost is as follows.
  - For each variable $x_i$, $c_{x_i}(o^+) = c_{x_i}(o^-) = 0$, and $c_{x_i}(o^0) = N_1$. As a result, an optimal mechanism never assigns $o^0$ to a variable.
  - For each positive literal $\ell^+$, $c_{\ell^+}(o^+) = N_2$, $c_{\ell^+}(o^0) = 0$, and $c_{\ell^+}(o^-) = N_1$. For each negative literal $\ell^-$, $c_{\ell^-}(o^+) = N_1$, $c_{\ell^-}(o^0) = 0$, and $c_{\ell^-}(o^-) = N_2$. We will see later that for any variable $x_i$, an optimal mechanism assigns precisely one of its literals the outcome with cost $N_2$, and the other outcome $o^0$ with cost $0$.
  - For each clause $C_j$, $c_{C_j}(o^+) = c_{C_j}(o^-) = 0$, and $c_{C_j}(o^0) = 1$.
- The types' utility functions are as follows.
  - For each variable $x_i$, $u_{x_i}(o^+) = u_{x_i}(o^-) = u_{x_i}(o^0)$. Note that the numerical values of the utility functions do not matter for deterministic mechanisms.
  - For each positive literal $\ell^+$, $u_{\ell^+}(o^+) > u_{\ell^+}(o^0) > u_{\ell^+}(o^-)$. For each negative literal $\ell^-$, $u_{\ell^-}(o^+) < u_{\ell^-}(o^0) < u_{\ell^-}(o^-)$.
  - For each clause $C_j$, $u_{C_j}(o^0) > u_{C_j}(o^+) = u_{C_j}(o^-)$.
- The reporting structure $R$ is as follows.
  - Each variable $x_i$ can only report itself.
  - Each literal $\ell$ can report itself or the variable it corresponds to.
  - Each clause $C_j$ can report itself, any literal $\ell \in C_j$ contained in the clause, or the variable $\ell$ corresponds to.

Now consider the structure of optimal deterministic mechanisms. For each variable $x_i$, an optimal mechanism assigns either $o^+$ or $o^-$. Moreover, for $x_i$'s two literals, if $x_i$ is assigned $o^+$ (resp. $o^-$), then the mechanism always assigns $x_i^+$ $o^+$ (resp. $o^0$) and $x_i^-$ $o^0$ (resp. $o^-$). One may check this is the only way to minimize cost subject to incentive compatibility. So conceptually, the mechanism chooses exactly one value for each variable, where assigning $o^0$ to $x_i^+$ (resp. $x_i^-$) corresponds to choosing value $1$ (resp. $0$) for $x_i$.

For each clause $C_j$, if any of the literals contained in $C_j$ is chosen (i.e., is assigned outcome $o^0$), then to prevent $C_j$ from misreporting that literal, the mechanism must assign $C_j$ outcome $o^0$, at a cost of $1$. This corresponds to the case where the clause is satisfied. Otherwise, if none of the literals in $C_j$ is chosen, the mechanism assigns either $o^+$ or $o^-$ to $C_j$, at a cost of $0$. The total cost of the mechanism is then $nN_2 + k$, where $k$ is the number of clauses satisfied. This encodes precisely the MinSAT instance. □

*Proof of Theorem 3.3.* First consider the runtime of Algorithm 3.1. The bottleneck is finding an $s$-$t$ min-cut on the graph $G$, which has $mn+2$ vertices and at most $mn^2+(m+1)n$ edges. Therefore, it is sufficient to show that one can replace the infinite capacities with capacity $W+1$.

We first prove that any horizontal edge with capacity $W+1$ does not belong to any min-cut. Suppose in some min-cut, for some $j \in [m]$, a horizontal edge from $(i_1, o_j) \in S$ to $(i_2, o_j) \notin S$ is cut. We argue that including all out-neighbors of $(i_1, o_j)$ through horizontal edges into $S$ strictly decreases the capacity of the cut. For each of these horizontal out-neighbors, by including it in $S$, we decrease the cut value by $W+1$ (from one horizontal edge), and possibly incur an additional cost from the edge between that neighbor and its vertical out-neighbor, whose capacity is at most $W$. Because we take the transitive closure of $R$, the newly included vertices do not have any horizontal out-neighbor out of $S$, so the total cost decreases at least by 1. A similar argument shows that edges leaving $S$ can be replaced to have capacity $W+1$ as well.

Now we move on to proving the correctness of Algorithm 3.1. We assume the infinite-capacity edges still have capacity $\infty$ (rather than $W+1$), which simplifies our argument. Observe that with infinite capacities, taking the transitive closure of $R$ in Line 3-5 of Algorithm 3.1 makes no difference. We prove the correctness for the algorithm without this step.

The argument consists of two parts. First we show there is a one-to-one correspondence between all finite-capacity *downward-closed* $s$-$t$ cuts and all deterministic truthful mechanisms, where the capacity of the cut is the same as the cost of the mechanism. We then show that taking the *downward closure* of any cut does not increase its capacity, and as a result, we only need to consider downward-closed cuts. These two claims together imply the correctness of Algorithm 3.1.

Formally, a cut $(S, \overline{S})$ is downward closed, if for any $i \in \Theta$ and $1 \le j_1 < j_2 \le m$,

$$(i, o_{j_2}) \in S \implies (i, o_{j_1}) \in S.$$

Fix a downward closed cut $(S, \overline{S})$, we construct a mechanism $M : \Theta \to \mathcal{O}$ in the same way as in Line 17-19 of Algorithm 3.1. That is, for all $i \in \Theta$,

$$M(i) = \max\{o_{j'} \in \mathcal{O} \mid (i, o_{j'}) \in S\}.$$

The one-to-one correspondence follows immediately from the definition of $M$. We now argue $(S, \overline{S})$ has finite capacity iff $M$ is truthful. Notice that $(S, \overline{S})$ has finite capacity iff no horizontal edge is cut, i.e., iff $M(i_1) \ge M(i_2)$ for all $(i_1, i_2) \in R$, which is precisely the condition for the truthfulness of $M$. Moreover, whenever $(S, \overline{S})$ has finite capacity, the capacity is equal to the cost of the truthful mechanism $M$.

Now we prove the second claim, i.e., taking the downward closure does not increase the capacity of the cut. We first define the downward closure. Given any $s$-$t$ cut $(S, \overline{S})$, the downward closure $(C(S), \overline{C(S)})$ is defined such that for all $i \in \Theta$,

$$(i, o_j) \in C(S) \iff j \le \max\{j' \in [m] \mid (i, o_{j'}) \in S\}.$$

We show below that the capacity of $C(S)$ is no larger than that of $S$.

If some horizontal edge is cut in $S$, then the cut has capacity $\infty$ and the claim is trivial. Suppose no horizontal edge is cut in $S$. Because the set of vertical edges cut in $C(S)$ is a subset of

those cut in $S$, we only need to show that no horizontal edge is cut in $C(S)$. Suppose for contradiction that some horizontal edges are cut in $C(S)$. Let $i_1, i_2 \in \Theta$ and $j \in [m]$ be such that $e = ((i_1, o_j), (i_2, o_j))$ is one of the highest horizontal edges being cut in $C(S)$. By the choice of $e$, it must be the case that $(i_1, o_j) \in S$, and since $S \subseteq C(S)$, we have $(i_2, o_j) \notin S$. Therefore, the same edge $e$ is also cut in $S$, which leads to a contradiction. □

*Proof of Lemma 3.1.* We prove the lemma by construction. Let $M$ be any (possibly randomized) optimal truthful mechanism. We construct a deterministic truthful mechanism from $M$ whose cost is no larger than that of $M$.

First we show it suffices for $M$ to randomize between only two consecutive outcomes for each type $i$. Let $p_i(o_j)$ be the probability that type $i$ receives outcome $o_j$. Suppose for some type $i$, there exist $j_1, j_2 \in [m]$, where $j_2 - j_1 > 1$, $p_i(o_{j_1}) > 0$, and $p_i(o_{j_2}) > 0$. We argue that one can move probability mass from $o_{j_1}$ and $o_{j_2}$ to $o_{j_3}$, where $j_3 = j_1 + 1$ lies between $j_1$ and $j_2$, without violating truthfulness or increasing the total cost.

Let $0 < \alpha < 1$ be such that $o_{j_3} = \alpha o_{j_1} + (1-\alpha) o_{j_2}$. Without loss of generality suppose $p_i(o_{j_1})/\alpha \le p_i(o_{j_2})/(1-\alpha)$. For brevity let $p = p_i(o_{j_1})$. We show that the following operation achieves the above goal.

- Decrease $p_i(o_{j_1})$ by $p$.
- Decrease $p_i(o_{j_2})$ by $(1-\alpha) \cdot p/\alpha \le p_i(o_{j_2})$.
- Increase $p_i(o_{j_3})$ by $p/\alpha$.

Observe that (1) after the operation, the probabilities of each outcome still sum to 1, and (2) type $i$ receives exactly the same expected utility. The principal's cost changes by

$$
\begin{aligned}
&\frac{p}{\alpha} \cdot c_i(o_{j_3}) - p \cdot c_i(o_{j_1}) - \frac{(1-\alpha)\cdot p}{\alpha} \cdot c_i(o_{j_2}) \\
&= \frac{p}{\alpha} \cdot \left(c_i(o_{j_3}) - \alpha \cdot c_i(o_{j_1}) - (1-\alpha) \cdot c_i(o_{j_2})\right) \\
&= \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - \alpha \cdot c_i^\ell(o_{j_1}) - (1-\alpha) \cdot c_i^\ell(o_{j_2})\right) && (c_i^\ell \text{ extends } c_i) \\
&\le \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - c_i^\ell(\alpha \cdot o_{j_1} + (1-\alpha) \cdot o_{j_2})\right) && (\text{convexity of } c_i^\ell) \\
&= \frac{p}{\alpha} \cdot \left(c_i^\ell(o_{j_3}) - c_i^\ell(o_{j_3})\right) = 0. && (o_{j_3} = \alpha \cdot o_{j_1} + (1-\alpha) \cdot o_{j_2})
\end{aligned}
$$

In other words, the total cost does not increase.

We then apply the above operation in the following way. Fix $i$, and let $j_- = \min\{j \mid p_i(o_j) > 0\}$, and $j_+ = \max\{j \mid p_i(o_j) > 0\}$. As long as $j_+ - j_- > 1$, apply the operation to $i$, $j_-$ and $j_+$. Observe that each time we apply the operation, $j_+ - j_-$ decreases by at least 1, so eventually we must stop and $j_+ - j_- \le 1$. Performing this for each $i$ yields a mechanism which randomizes only between two consecutive outcomes for each type, without increasing the total cost. Without loss of generality, from now on, we assume $M$ has this property.

Now we show there is a way to round $M$, producing a distribution over deterministic truthful mechanisms, such that the expected cost of this distribution is precisely the cost of $M$. As a result,

there exists one mechanism in the support of the distribution, whose cost is upper bounded by that of $M$, which is our desired deterministic truthful mechanism.

For each type $i$, let $j_i = \min\{j \mid p_i(j) > 0\}$ and $\alpha_i = p_i(j_i + 1)$. Note that $0 \leq \alpha_i < 1$ for all $i$. For any $r \in [0, 1]$, let $M_r$ be the deterministic mechanism defined such that for each type $i$,

$$M_r(i) = \begin{cases} o_{j_i+1}, & r \leq \alpha_i, \\ o_{j_i}, & \text{otherwise.} \end{cases}.$$

We first argue that $M_r$ is truthful for any $r \in [0, 1]$. Fix any pair $(i_1, i_2) \in R$. Given that $M$ itself is truthful, we proceed by considering the following two cases.

- $j_{i_1} > j_{i_2}$. In such cases, we always have $M_r(i_1) \geq M_r(i_2)$, so $i_1$ has no incentive to report $i_2$.
- $j_{i_1} = j_{i_2}$ and $\alpha_{i_1} \geq \alpha_{i_2}$. For any $r \in [0, 1]$, we have

  $$r \leq \alpha_{i_1} \impliedby r \leq \alpha_{i_2}.$$

  So again, regardless of $r$, $M_r(i_1) \geq M_r(i_2)$.

Applying the above argument to each pair $(i_1, i_2) \in R$ establishes the truthfulness of $M_r$ for any $r \in [0, 1]$.

Now consider the distribution over deterministic mechanisms $M_r$ when $r$ is uniformly distributed over $[0, 1]$. We show that the expected cost of $M_r$ is equal to the cost of $M$:

$$\begin{aligned} \mathbb{E}_r[c(M_r)] &= \sum_{i \in \Theta} \left(\Pr[r \leq \alpha_i] \cdot c_i(o_{j_i+1}) + \Pr[r > \alpha_i] \cdot c_i(o_{j_i})\right) \\ &= \sum_{i \in \Theta} (\alpha_i \cdot c_i(o_{j_i+1}) + (1 - \alpha_i) \cdot c_i(o_{j_i})) \\ &= \sum_{i \in \Theta} (p_i(j_i + 1) \cdot c_i(o_{j_i+1}) + p_i(j_i) \cdot c_i(o_{j_i})) \\ &= \sum_{i \in \Theta} \mathbb{E}_M[c_i(M(i))] \\ &= c(M), \end{aligned}$$

which concludes the proof. □

*Proof of Theorem 3.4.* We prove the correctness first. Observe that the problem of finding a (ran-

domized) optimal mechanism can be written as the following linear program.

$$\begin{aligned}
\min \quad & \sum_{i\in\Theta=[n],j\in[m]} p_{i,j}\cdot c_i(o_j) & \\
\text{subject to} \quad & u_i = \sum_j p_{i,j}\cdot o_j & \forall i\in\Theta \\
& u_{i_1}\geq u_{i_2} & \forall (i_1,i_2)\in R \\
& \sum_j p_{i,j} = 1 & \forall i\in\Theta \\
& p_{i,j}\geq 0 & \forall i\in\Theta, j\in[m].
\end{aligned}$$

Here, $u_i$ is the expected utility of type $i$, and $p_{i,j}$ is the probability of assigning type $i$ outcome $o_j$. This is not the most succinct LP formulation of the problem, but it capture the structure of the problem in a way that is more useful for our analysis.

Now fix $\{u_i\}_i$ and consider the optimal choice of $\{p_{i,j}\}_{i,j}$. This can be solved separately for each type $i$, by considering the following linear program (with the additional constraints that $\{p_{i,j}\}_j$ are nonnegative and sum up to 1 for all $i$).

$$\begin{aligned}
\min \quad & \sum_j p_{i,j}\cdot c_i(o_j) \\
\text{s.t.} \quad & \sum_j p_{i,j}\cdot o_j = u_i.
\end{aligned}$$

This is precisely evaluating the convex envelope $c_i^-$ of $c_i$ at $u_i\in[o_1,o_m]$. Consequently, the problem of finding a (randomized) optimal mechanism can be rewritten as the following convex program.

$$\begin{aligned}
\min \quad & \sum_{i\in\Theta} c_i^-(u_i) & \\
\text{s.t.} \quad & u_{i_1}\leq u_{i_2}, & \forall (i_1,i_2)\in R, \\
& u_i\in[o_1,o_m], & \forall i\in\Theta.
\end{aligned}$$

Now observe that the reformulated program cannot distinguish between $\{c_i\}_{i\in\Theta}$ and $\{\widehat{c_i}\}_{i\in\Theta}$, where $\widehat{c_i}$ is $c_i^-$ restricted to $\mathcal{O}$ as in Algorithm 3.2 — the two cost functions simply induce exactly the same program. Moreover, observe that the newly constructed cost function $\{\widehat{c_i}\}_{i\in\Theta}$ is convex, according to Definition 3.1. Given this convexity, Lemma 3.1 implies that there exists a deterministic mechanism for $\{\widehat{c_i}\}_i$ which is optimal. In other words, there exists an optimal solution $\{u_i\}_i$ to the reformulated program in which $u_i\in\mathcal{O}$ for each $i\in\Theta$. Algorithm 3.2 finds such a solution $\{u_i\}_i$ by calling Algorithm 3.1.

Now the only problem left is to recover $\{p_{i,j}\}_j$ from $u_i$ for each type $i$. This is done in Line 6 of Algorithm 3.2. Since the output mechanism $M$ implements $\{u_i\}_i$ in an optimal way, it is a truthful mechanism that minimizes the principal's cost. This establishes the correctness of Algorithm 3.2.

Now we consider the time complexity. Compared to Algorithm 3.1, the additional steps in Algorithm 3.2 include (1) computing $\{\widehat{c_i}\}_i$ in Line 2, and (2) interpreting $\widehat{M}(i)$ as an optimal

convex combination of outcomes in Line 6. We will show that both operations can be done in time $O(mn)$, i.e., linear in the size of the input. The time complexity of Algorithm 3.2 then follows immediately.

For computing the convex envelope of $c_i$, one may use the classical algorithm by Andrew [2], which scans $\mathcal{O}$ from left to right, and maintains a stack containing the partial convex envelope of $c_i$ on $\{o_1, \ldots, o_j\}$ for every $j \in [m]$. The algorithm runs in time $O(m)$.

Once we know $\{\widehat{c}_i\}$, to find an optimal convex combination for a target utility $u_i$, we first find in time $O(m)$ the largest integer $\ell \in [m]$ such that $o_\ell \leq u_i$ and $c_i(o_\ell) = \widehat{c}_i(o_\ell)$, and the smallest integer $r \in [m]$ such that $o_r \geq u_i$ and $c_i(o_r) = \widehat{c}_i(o_r)$. If $\ell = r$, then we output $o_\ell = o_r$. Otherwise, there is a unique $\alpha \in (0, 1)$ such that randomizing between $o_\ell$ and $o_r$ gives expectation $\alpha \cdot o_\ell + (1-\alpha) \cdot o_r = u_i$. The convex envelope is linear between $o_l$ and $o_r$, and hence $\alpha \cdot c_i(o_\ell) + (1-\alpha) \cdot c_i(o_r) = \widehat{c}_i(u_i)$. Then, we can set $M(i)$ to $o_\ell$ with probability $\alpha$ and to $o_r$ with probability $(1 - \alpha)$. Performing the above for every type $i$ takes $O(mn)$ time. □

# Chapter 4

# Classification with Strategically Withheld Data

## 4.1 Introduction

In Chapters 2 and 3, we have established a series of theoretical results for strategic classification. Now we investigate the applied aspects of strategic classification in this chapter. Consider the following setting: Applicants to most colleges in the US are required to submit their scores for at least one of the SAT and the ACT. Both tests are more or less equally popular, with close to two million taking each in 2016 [1]. Applicants usually take one of these two tests – whichever works to their advantage.[1] However, given the growing competitiveness of college admissions, many applicants now take both tests and then strategically decide whether to drop one of the scores (if they think it will hurt their application) or report both.[2] The key issue here is that it is impossible to distinguish between an applicant who takes both tests but reports only one, and an applicant that takes only one test—for example because the applicant simply took the one required by her school, the dates for the other test did not work with her schedule, or for other reasons that are not strategic in nature.[3]

Say a college wants to take a principled machine learning approach to making admission decisions based on the scores from these two tests. For simplicity, assume no other information is available. Assume that the college has enough historical examples that contain the scores of individuals (on whichever tests are taken, truthfully reported) along with the corresponding ideal (binary) admission decisions.[4] Based on this data, the college has to choose a decision function that determines which future applicants are accepted. If this function is known to the applicants, they are bound to strategize and use their knowledge of the decision function to decide the scores they report.[4] How can the classifier be trained to handle strategic reporting of scores at prediction time?

[1] `https://www.princetonreview.com/college/sat-act`
[2] `https://blog.collegevine.com/should-you-submit-your-sat-act-scores/`
[3] `https://blog.prepscholar.com/do-you-need-to-take-both-the-act-and-sat`
[4] We make these assumptions more generally throughout the chapter.

To see the intricacies of this problem, let us consider a simple example.

**Example 4.1.** Say the scores for each of the two tests (SAT and ACT) take one of two values: $h$ (for high) or $l$ (for low). Let $*$ denote a missing value. Then there are eight possible inputs (excluding $(*,*)$ since at least one score is required): $(h,h)$, $(h,l)$, $(l,h)$, $(l,l)$, $(h,*)$, $(*,h)$, $(l,*)$ and $(*,l)$. Assume the natural distribution (without any withholding) over these inputs is known, and so are the conditional probabilities of the label $Y \in \{0,1\}$, as shown below:

| $X$ | $(h,h)$ | $(h,l)$ | $(l,h)$ | $(l,l)$ | $(h,*)$ | $(*,h)$ | $(l,*)$ | $(*,l)$ |
|---|---|---|---|---|---|---|---|---|
| $Pr(X)$ | $1/8$ | $1/8$ | $1/8$ | $1/8$ | $1/8$ | $1/8$ | $1/8$ | $1/8$ |
| $Pr(Y=1 \mid X)$ | 0.9 | 0.7 | 0.3 | 0.1 | 0.6 | 0.6 | 0.2 | 0.2 |
| $Pr(Y=0 \mid X)$ | 0.1 | 0.3 | 0.7 | 0.9 | 0.4 | 0.4 | 0.8 | 0.8 |

Table 4.1: True distribution of inputs and targets

Assume $Y=1$ is the more desirable "accept" decision. Then, ideally, we would like to predict $\widehat{Y}=1$ whenever $X \in \{(h,h),(h,l),(h,*),(*,h)\}$. However, the strategic reporting of scores at prediction time effectively means, for example, that an input $(*,h)$ cannot be assigned the accept decision of $\widehat{Y}=1$ unless the same is done for $(l,h)$ as well; otherwise, someone with $(l,h)$ would simply not report the first test, thereby misreporting $(*,h)$ and being accepted. Taking this into account, the classifier with minimum error is given by $\widehat{Y}=1$ whenever $X \in \{(h,h),(h,l),(h,*)\}$.

There are many other settings where a similar problem arises. Many law schools now allow applicants to choose between the GRE and the traditional LSAT.[5] Recently, as a result of the COVID-19 pandemic, universities have implemented optional pass/fail policies, where students can choose to take some or all of their courses for pass/fail credit, as opposed to a standard letter grade that influences their GPA. They are often able to decide the status after already knowing their performance in the course. For credit scoring, some individuals might not report some of their information, especially if it is not mandatory by law [44].

The ability of strategic agents to withhold some of their features at prediction time poses a challenge only when the data used to train the classifier has some naturally missing components to begin with. For if not, the *principal* – e.g., the entity deciding on admissions – can reject all agents that withhold any of their features, thereby forcing them to reveal all features. We focus on how a principal can best train classifiers that are robust even when there is strategic withholding of data by agents. Our methods produce classifiers that eliminate the incentive for agents to withhold data.

**Our contributions** We now describe the key questions we are facing, and how we answer them. Our model is described formally in *Section 4.2*. All proofs are defered to the end of the chapter.

*If the true input distribution is known, can we compute the optimal classifier? (Section 4.3)* We answer this question in the affirmative by showing that the problem of computing the optimal classifier (Theorem 4.1) in this setting reduces to the classical Min-cut problem [29]. This analysis gives us the MINCUT classifier (a special case of the general algorithm presented in Chapter 3), which can be computed on the empirical distribution, estimated using whatever data is available. However, since it can potentially give complex decision boundaries, it might not generalize well.

[5] `https://www.ets.org/gre/revised_general/about/law/`

*Are there simpler classifiers that are robust to strategic withholding of features? (Section 4.4)* We first characterize the structure of classifiers that are "truthful", i.e., give no incentive to strategically hide features at prediction time (Theorem 4.2). Using this characterization, we devise a hill-climbing procedure (HC) to train a hierarchical ensemble of out-of-the-box classifiers and show that the procedure converges (Theorem 4.4) as long as we have black-box access to an agnostic learning oracle. We also analytically bound the generalization error of HC (Theorem 4.3). The ensemble of HC can be populated with any of the commonly used classifiers such as logistic regression, ANNs, etc.

Another truthful classifier we present is a modification of Logistic Regression. This method, called IC-LR (Incentive Compatible Logistic Regression), works by encoding all features with positive values, and using positive regression coefficients – whereby it is in every agent's best interest to report all features truthfully. IC-LR uses Projected Gradient Descent for its training. The advantage of this method is that it can be directly to a large number of features.

*How do our methods perform on real data sets? (Section 4.6)* We conduct experiments on several real-world data sets to test the performance of our methods, comparing them to each other, as well as to other methods that handle missing data but ignore the strategic aspect of the problem. We see that our methods perform well overall, and uncover some interesting insights on their relative performance:

1. When the number of features is small, HC is the most reliable across the board.
2. When the number of features is small, and many of them are discrete/categorical (or suitably discretized), MINCUT and IC-LR perform better.
3. If a large number of features must be used, IC-LR gives the best performance, although HC performs reasonably well with some simple feature selection techniques.

## 4.2 Preliminaries

**Model with strategically withheld features:** We have an input space $\mathcal{X}$, a label space $\mathcal{Y} = \{0, 1\}$, and a distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$ which models the population. A classifier $f : \mathcal{X} \to \mathcal{Y}$ maps a combination of features to a label. Let $F = [k] = \{1, \ldots, k\}$ be the set of features, each of which a data point may or may not have. For $x \in \mathcal{X}$, let $x_i$ denote the value of its $i$-th feature ($x_i = *$ if $x$ does not have feature $i \in [k]$). For any $S \subseteq [k]$, define $x|_S$ to be the projection of $x$ onto $S$ (i.e., retain features in $S$ and drop those not in $S$):

$$(x|_S)_i = \begin{cases} x_i, & \text{if } i \in S \\ *, & \text{otherwise.} \end{cases}$$

We assume that data can be strategically manipulated at prediction (test) time in the following way: an agent whose true data point is $x$ can report any other data point $x'$ such that $x|_S = x'$ for some $S \subseteq [k]$. We use $\to$ to denote the relation between any such pair $x, x'$ ($x \to x' \iff \exists S \subseteq [k] : x|_S = x'$). Note that $\to$ is transitive, i.e., for any $x_1, x_2, x_3 \in \mathcal{X}$, $x_1 \to x_2$ and $x_2 \to x_3 \implies x_1 \to x_3$.

We assume agents prefer label 1 to 0: in response to a classifier $f$, an agent with data point $x$ will always withhold [6] features to receive label 1 if possible, i.e., the agent will report $x' \in \text{argmax}_{x'':x\to x''} f(x'')$. Incorporating such strategic behavior into the loss of a classifier $f$, we get

$$\ell_{\mathcal{D}}(f) = \Pr_{(x,y)\sim\mathcal{D}}\left[y \neq \max_{x':x\to x'} f(x')\right].$$

**Truthful classifiers** We will also be interested in *truthful* classifiers, which provably eliminate incentives for such strategic manipulation. A classifier $f$ is *truthful* if for any $x, x' \in \mathcal{X}$ where $x \to x'$, $f(x) \geq f(x')$. In other words, not withholding any features is always an optimal way to respond to a truthful classifier. As a result, the loss of any truthful classifier $f$ in the presence of strategically withheld features has the standard form: $\ell_{\mathcal{D}}(f) = \Pr_{(x,y)\sim\mathcal{D}}[f(x) \neq y]$.

Note that the so-called Revelation Principle – which states that in the presence of strategic behavior, any classifier $f$ is equivalent to a truthful classifier $f'$ – holds in this case because the reporting structure is transitive.[7] In other words, we are guaranteed that, for any classifier $f$, there exists a truthful classifier $f'$, such that for any $x \in \mathcal{X}$, $\max_{x':x\to x'} f(x') = f'(x)$. Therefore, we focus on truthful classifiers in our model, without loss of generality.

## 4.3 The MINCUT Classifier

We first present a method for computing an optimal classifier *when the input distribution is fully known*. Assuming $\mathcal{X}$ is finite, our goal is to characterize a classifier $f^*$ which minimizes the loss $\ell_{\mathcal{D}}(.)$, for a known input distribution $\mathcal{D}$. As shorthand, define, for all $x \in \mathcal{X}$,

**Definition 4.1.** $\mathcal{D}^+(x) \coloneqq \Pr_{(x',y')\sim\mathcal{D}}[x' = x \wedge y' = 1]$,
$\mathcal{D}^-(x) \coloneqq \Pr_{(x',y')\sim\mathcal{D}}[x' = x \wedge y' = 0]$.

The basic idea here is simple: to partition $\mathcal{X}$ into two sides, one labeled 1 and the other 0, where the error accrued for each $x \in \mathcal{X}$ is given by $\mathcal{D}^-(x)$ or $\mathcal{D}^+(x)$, according as $x$ is labeled 1 or 0. Such a partition should crucially respect the constraints imposed by the strategic behavior of agents : if $x \to x'$, then either $x$ is labeled 1 or $x'$ is labeled 0.

**Definition 4.2.** Given $\mathcal{X}$ and $\mathcal{D}$, let $G(\mathcal{D}, \mathcal{X})$ be a directed capacitated graph with vertices $V = \mathcal{X} \cup \{s, t\}$, where the edges $E$ and edge capacities $u$ are defined as follows:

- For each $x \in \mathcal{X}$, there are edges $(s, x)$ and $(x, t)$ in $E$, with capacities $u(s, x) = \mathcal{D}^-(x)$ and $u(x, t) = \mathcal{D}^+(x)$.
- For all pairs $x, x' \in \mathcal{X}$ such that $x \to x'$, there is an edge $(x, x') \in E$ with capacity $u(x, x') = \infty$.

In terms of the graph defined above, computing the optimal classifier $f^*$ we seek is equivalent to finding a minimum $s$-$t$ cut on $G(\mathcal{D}, \mathcal{X})$. The intuition is that the edges from $s$ and to $t$ reflect the value gained from labeling an example 0 or 1, respectively; one of the edges must be cut, reflecting

[6] In practice, $f$ might not be perfectly known, and agents might not be able to best respond. This problem does not arise for our methods, since they are truthul. For other classifiers, their accuracy may go up or down if agents fail to best-respond; but the assumption that agents best-respond is common in many such contexts.

[7] More details, including a formal proof, can be found in the Supplement of [66].

the loss of not assigning it to the corresponding side. Moreover, if $x \to x'$, then the corresponding edge with infinite capacity prevents the assigning of 0 to $x$ and 1 to $x'$.

**Theorem 4.1.** *If $(S, \bar{S})$ is a minimum $s$-$t$ cut of $G(\mathcal{D}, \mathcal{X})$ (where $S$ is on the same side as $s$), then for the classifier $f^*(x) \coloneqq \mathbb{1}(x \in \bar{S})$, we have $\ell_{\mathcal{D}}(f^*) = \min_f \ell_{\mathcal{D}}(f)$.*

We note that, consequently, the optimal classifier can be computed in $\mathrm{poly}(|\mathcal{X}|)$ time. In practice, it is natural to expect that we do not know $\mathcal{D}$ exactly, but have a finite number of samples from it. A more practical option is to apply Theorem 4.1 to the empirical distribution induced by the samples observed, and hope for the classifier computed from that to generalize to the true population distribution $\mathcal{D}$.

**Implementing** MINCUT Given a set $\widehat{\mathcal{X}}$ of $m$ i.i.d. samples from $\mathcal{D}$, let $\widehat{\mathcal{D}}$ be the corresponding empirical distribution over $\widehat{\mathcal{X}}$, and $\bar{\mathcal{X}} \coloneqq \widehat{\mathcal{X}} \cup \{x' : x' \to x, \exists x \in \widehat{\mathcal{X}}\}$. The MINCUT classifier is then obtained by applying Theorem 4.1 to $G(\widehat{\mathcal{D}}, \widehat{\mathcal{X}})$, and extending it to $\bar{\mathcal{X}}$ as and when required. Here, note that MINCUT runs in time $\mathrm{poly}(m)$ (and not $\mathrm{poly}(|\mathcal{X}|)$), since $G(\widehat{\mathcal{D}}, \widehat{\mathcal{X}})$ has $m$ nodes, and checking if a test point is in $\bar{\mathcal{X}}$ takes $\mathrm{poly}(m)$ time.

In light of traditional wisdom, the smaller $m$ is relative to $\mathcal{X}$, the larger the generalization error will be. While not attempting a theoretical analysis in this regard, we note that when $\mathcal{X}$ is large, the generalization error can be extremely large (see Example 2 in the Supplement of [66]). The reason for this is two-fold:

1. MINCUT can give complicated decision boundaries.
2. MINCUT is indecisive on samples not in $\bar{\mathcal{X}}$.[8]

Therefore, a suitable discretization of features is sometimes useful (see Section 4.6). Note that MINCUT is truthful, by virtue of the infinite capacity edges in Definition 4.2.

## 4.4 Truthful Classifiers and HILL-CLIMBING

The other drawback of MINCUT, related to the issue of generalization just discussed, is that it can be hard to interpret meaningfully in a practical setting. In this section, we devise a simpler alternative called HILL-CLIMBING. To help introduce this algorithm, we first present a characterization of truthful classifiers in our setting, since we can limit our focus to them without loss of generality (as discussed in Section 4.2). For shorthand, we use the following definition:

**Definition 4.3** ($F'$-classifier)**.** For a subset of features $F' \subseteq F$, a classifier $f$ is said to be an $F'$-classifier if for all $x \in \mathcal{X}$, we have $f(x) = f(x|_{F'})$, and if there exists $i \in F'$ such that $x_i = *$, then $f(x) = 0$.

In other words, an $F'$-classifier depends only on the features in $F'$, rejecting all $x$ where any of these is empty. We collect many such classifiers into an ensemble as follows:

**Definition 4.4** (MAX Ensemble)**.** For a collection of classifiers $\mathcal{C} = \{f_j\}$, its MAX Ensemble classifier is given by $\mathrm{MAX}_{\mathcal{C}}(.) \coloneqq \max_j f_j(.)$.

[8]This is more likely to happen when using a large number of features.

This is equivalent to getting each agent to pick the most favorable classifier from among those in $\{f_j\}$. Now we have the following characterization of truthful classifiers:

**Theorem 4.2.** *A classifier $f$ is truthful iff $f(.) = \textsc{Max}_{\mathcal{C}}(.)$ for a collection of classifiers $\mathcal{C} = \{f_j\}$ such that, for some $\{F_j\} \subseteq 2^F$, each $f_j$ is an $F_j$-classifier .*

For any truthful classifier $f$, we seek to bound the gap between its population loss $\ell_{\mathcal{D}}(f)$ and its empirical loss on samples in $\widehat{\mathcal{X}}$ defined by $\ell_{\widehat{\mathcal{X}}}(f) \coloneqq \frac{1}{m}\sum_{i\in[m]} |f(x_i) - y_i|$. Before stating a theorem to this end, we define the following entities: Let $\mathcal{H}$ be a base hypothesis space over $\mathcal{X}$, and $n \in \{1, \ldots, 2^k\}$ be a parameter. Define $d \coloneqq d_{\text{VC}}(\mathcal{H})$, the VC dimension of $\mathcal{H}$, and $\bar{\mathcal{H}}$ is the set of all classifiers that can be written as the MAX Ensemble of $n$ classifiers in $\mathcal{H}$.

**Theorem 4.3.** *Let $\widehat{\mathcal{X}} = \{(x_i, y_i)\}_{i\in[m]}$ be $m$ i.i.d. samples from $\mathcal{D}$. For any $f \in \bar{\mathcal{H}}$, for any $\delta > 0$, with probability at least $1 - \delta$, we have $\ell_{\mathcal{D}}(f) \leq \ell_{\widehat{\mathcal{X}}}(f) + O\left(\sqrt{\frac{dn\cdot\log dn\cdot\log m+\log(1/\delta)}{m}}\right)$.*

It is easy to see that for any of the commonly used hypothesis spaces – say $\mathcal{H}$ consists of linear hypotheses – if a truthful classifier $f$ is in $\mathcal{H}$, then so are the components of the MAX Ensemble version of $f$ as in Theorem 4.2. We have, however, stated Theorem 4.3 in slightly more general terms.

**The HILL-CLIMBING classifier** We now present a hill-climbing approach with provable convergence and generalization guarantees. The HILL-CLIMBING classifier (henceforth HC) is of the same form as given by the characterization of truthful classifiers in Theorem 4.2.[9] Intuitively, the approach works by considering a hierarchy of classifiers, organized by the features involved. For example, consider a setting with $k = 3$ features. We make a choice as to what classifiers we use — say $f_1$ for input of the form $(x_1, *, *)$, $f_2$ for input of the form $(x_1, x_2, *)$, and $f_3$ for input of the form $(x_1, x_2, x_3)$. Any agent with features 1 and 2 (but not 3), for example, should be able to report both features so as to be classified by $f_2$, or feature 2 to be classified by $f_1$ instead. So in effect, assuming full knowledge of the classifiers, each agent can check all of the classifiers and choose the most favorable one. Without loss of generality, we assume that when a data point does not have all the features required by a classifier, it is automatically rejected.

In short, HC (defined formally in Algorithm 4.1) works as follows: first choose a hypothesis space $\mathcal{H}$, in order for Theorem 4.3 to apply. Then select $n$ subsets of $F$ (where $n$ is a parameter), say $F_1, F_2, \ldots, F_n$. For each $F_j$, we learn a $F_j$-classifier, say $f_j$, from among those in $H$. Start by initializing these classifiers to any suitable $\{f_1^0, \ldots, f_n^0\}$. In each iterative step, each of the subclassifiers is updated to minimize the empirical loss on the samples that are rejected by all other classifiers. We next show that such an update procedure always converges. To do so, as far as our theoretical analysis goes, we assume we have black-box access to an agnostic learning oracle (Line 6 in Algorithm 4.1). After convergence, the HC classifier is obtained as the MAX Ensemble of these classifiers. The generalization guarantee of Theorem 4.3 applies directly to the HC classifier.

**Theorem 4.4.** *Algorithm 4.1 converges.*

[9] And, therefore, is truthful, and inherits Theorem 4.3.

**Algorithm 4.1:** HILL-CLIMBING (HC) Classifier

**Input:** data set $\widehat{\mathcal{X}} = \{(x_i, y_i)\}_{i\in[m]}$, n subsets $F_1, F_2, \ldots, F_n$ of F.
Initialize: $t \leftarrow 0$, $\{f_1^0, \ldots, f_n^0\}$.
**while** $\Delta > 0$ **do**
  **for** $i = 1, 2, \ldots, n$ **do**
    $S_i \leftarrow \{(x, y) \in \widehat{\mathcal{X}} : f_j^t(x|_{F_j}) = 0, \forall j \neq i\}$.
    $f_i^{t+1} = \text{argmin}_{f\in\mathcal{H}} \sum_{(x,y)\in S_i} |f(x|_{F_i}) - y|$.
  **end for**
  $f^* \leftarrow \text{MAX}_{\{f_1^{t+1},\ldots,f_n^{t+1}\}}$; $\ell_t = \ell_{\widehat{\mathcal{X}}}(f^*)$
  $\Delta \leftarrow \ell_t - \ell_{t-1}$; $t \leftarrow t + 1$
**end while**
**Return:** $f^*$.

**Connection with** MINCUT The HC formulation given above can be thought of as a less complicated version of MINCUT: some of the edge constraints are ignored with respect to learning the individual classifiers, and are instead factored in via the MAX function. Say $F_1 \subset F_2$. For some $x$, it is possible that $f_1(x|_{F_1}) = 1$ and $f_2(x|_{F_1}) = 0$. In other words, the individual classifiers could violate the MINCUT constraints, in order to learn classification functions that generalize well individually, and also collectively thanks to the combined HC training procedure.

**Implementation** The set of classifiers $\{f_1, f_2, \ldots, f_n\}$ in HC can be populated with any standard out-of-the-box methods such as logistic regression or neural networks, the choice of which can influence the performance of $f$. In Section 4.6, we test HC with a few such options. The assumption of having access to an agnostic learning oracle does not play a crucial role in practice, with standard training methods performing well enough to ensure convergence.

Also, HC will converge in at most m (number of training examples) iterations, because in each iteration the number of correctly classified examples increases by at least one. (An iteration may need to train n individual classifiers.) This also means there is no difference between checking whether $\Delta > 0$ or $\Delta \geq 1/m$. In our experiments, we run HC using $\Delta \geq 10^{-4}$, and convergence is achieved pretty quickly (see the Supplement of [66] for exact details).

**Choosing subsets** Note that we are free to choose any $F_1, F_2, \ldots, F_n$ to define HC. Its generalization (via Theorem 4.3), will depend on the choice of $n$. As more and more subsets of features are included (and further binning them based on their values), HC starts behaving more and more like MINCUT. In addition, using a large number of subsets increases the computational complexity of HC. In practice, therefore, the number of subsets must be limited somehow – we find that some simple strategies like the following work reasonably well: (a) selecting a few valuable features and taking all subsets of those features, (b) taking all subsets of size smaller than a fixed number $k$, say $k = 2$. In many practical situations, a few features (possibly putting their values in just a few bins) are often enough to get close to optimal accuracy, also improving interpretability (e.g., see

Wang and Rudin [101] or Jung et al. [62]) The question of devising a more nuanced algorithm for selecting these subsets merits a separate investigation, and we leave this to future work.

## 4.5 Incentive-Compatible Logistic Regression

As we just mentioned, it is challenging to directly apply HC and MINCUT to a large number of features. As we will see, we can address this challenge in various ways to still get very strong performance with HC. Moreover, HC enjoys remarkable generality, generalization and convergence guarantees. Nevertheless, we would like to have an algorithm that tries to make use of all the available features, while still being truthful. In this section, we present such an approach, which, as we show later in Section 4.6, indeed performs comparably to – and in some cases better than – MINCUT and HC.

Below we present a simple and truthful learning algorithm, Incentive-Compatible Logistic Regression (IC-LR), which is a truthful variant of classical gradient-based algorithms for logistic regression. Recall that in logistic regression, the goal is to learn a set of coefficients $\{\beta_i\}$, one for each feature $i \in F$, as well as an intercept $\beta_0$, such that for each data point $(x, y)$, the predicted label $\hat{y}$ given by

$$\hat{y} = \mathbb{1}\left[\sigma(\beta_0 + \sum_{i \in F} x_i \cdot \beta_i) \geq 0.5\right]$$

fits $y$ as well as possible, where $\sigma(t) = 1/(1 + e^{-t})$ is the logistic function. Roughly speaking, IC-LR. (formally defined in Algorithm 4.2) works by restricting the coefficients $\{\beta_i\}$ in such a way that dropping a feature (i.e., setting $x_i$ to 0) can never make the predicted label larger. If, without loss of generality, all feature values $x_i$ are nonnegative (or suitably translated), then this is equivalent to: for each feature $i \in F$, the coefficient $\beta_i \geq 0$. IC-LR. enforces this nonnegativity constraint throughout the training procedure, by requiring a projection step after each gradient step, which projects the coefficients to the feasible nonnegative region by setting any negative coefficient to 0 (equivalently, an $\ell_1$ projection).

One potential issue with IC-LR is the following: if a certain feature $x_i \geq 0$ is negatively correlated with the positive classification label, then IC-LR is forced to ignore it (since it is constrained to use positive coefficients). To make good use of this feature, we can include an inverted copy $x_i' = \lambda - x_i$ (where $\lambda$ is chosen such that $x_i' \geq 0$). We could also choose an apt discretization of such features (using cross-validation) and translate the discretized bins into separate binary variables. Such a discretization can account for more complex forms of correlation, e.g., a certain feature's being too high or too low me makes the positive label likelier. In practice, we find that the latter method does better. If such transformations are undesirable, perhaps for reasons of complexity or interpretability, HC methods are a safer bet.

## 4.6 Evaluation

In this section, we show that, when strategic withholding is at play, MINCUT, HC and IC-LR perform well and provide a significant advantage over several out-of-the-box counterparts (that do

**Algorithm 4.2:** Incentive-Compatible Logistic Regression

**Input:** data set $\widehat{\mathcal{X}} = \{(x, y)\}$, learning rate $\{\eta_t\}$, $\delta \geq 0$.
Initialize: $t \leftarrow 0$, $\{\beta_0, \beta_1, \ldots, \beta_k\}$.
**while** $\Delta > \delta$ **do**
  $g_i \leftarrow 0$ for all $i \in \{0, 1, \ldots, k\}$
  **for** $(x, y) \in \widehat{\mathcal{X}}$ **do**
    $g_0 \leftarrow g_0 + \sigma\left(\beta_0 + \sum_{i \in F} x_i \cdot \beta_i\right) - y$
    **for** $i \in F$ **do**
      $g_i \leftarrow g_i + (\sigma\left(\beta_0 + \sum_{i \in F} x_i \cdot \beta_i\right) - y) \cdot x_i$
    **end for**
  **end for**
  $\forall i \in \{0, 1, \ldots, k\}$, $\beta_i \leftarrow \max\{\beta_i - \eta_t \cdot g_i, 0\}$
  $f^*(x) \coloneqq \mathbb{1}\left(\sigma\left(\beta_0 + \sum_{i \in F} \beta_i \cdot x_i\right) \geq 0.5\right)$
  $\ell_t = \ell_{\widehat{\mathcal{X}}}(f^*)$; $\Delta \leftarrow \ell_t - \ell_{t-1}$; $t \leftarrow t + 1$
**end while**
**Return:** $f^*$.

not account for strategic behavior).

**Datasets** Four credit approval datasets are obtained from the UCI repository [37], one each from Australia, Germany, Poland and Taiwan. As is common for credit approval datasets, they are imbalanced to various degrees. In order to demonstrate the performance of classifiers in a standard, controlled setting, we balance them by random undersampling. There is a dedicated community [18] that looks at the issue of imbalanced learning. We do not delve into these issues in this chapter, and evaluate our methods on both balanced and imbalanced datasets (see the Supplement of [66] for the latter). In addition, to demonstrate the challenge of high-dimensional data imposed on some of the classification methods, the experiments are run on the datasets (a) restricted to 4 features,[10] and (b) with all available features. The basic characteristics of the datasets are summarized in Table 4.2 – note that there is enough variation in terms of the types of features present. We then randomly remove a fraction $\epsilon = 0, 0.1, \ldots, 0.5$ of all feature values in each dataset to simulate data that is missing "naturally" – i.e., not due to strategic withholding.

**Testing** We test all methods under two ways of reporting: "truthful", i.e., all features are reported as is, and "strategic", i.e., some features might be withheld if it leads to a better outcome. We measure the test accuracy of each classifier, averaged over N=100 runs, with randomness over the undersampling and the data that is randomly chosen to be missing, to simulate data missing for non-strategic reasons. Other metrics, and details about implementing and training the classifiers, are discussed in the Supplement of [66]. It is important to note that for testing any method, we have to, in effect, compute the best response of each data point toward the classifier. Since the methods

[10] According to ANOVA F-value evaluated before dropping any feature values.

| Data set | Size | Total # of features | Size after balancing | Features after restriction |
|---|---|---|---|---|
| Australia | 690 | 15 | 614 | 2 num., 2 cat. |
| Germany | 1000 | 20 | 600 | 1 num., 3 cat. |
| Poland | 5910 | 64 | 820 | 4 num. |
| Taiwan | 30,000 | 23 | 13,272 | 4 ordinal |

Table 4.2: Data set summary statistics (num. = numerical, cat. = categorical)

we propose are truthful, this is a trivial task. But for other methods, this might not be easy, thereby limiting what baselines can be used.

**Classifiers** We evaluate our proposed methods, MINCUT, HC with logistic regression (HC (LR)) and neural networks (HC (ANN)) as subclassifiers, and incentive-compatible logistic regression (IC-LR), against several baseline methods.

First, they will be compared against three out-of-the-box baseline classifiers: logistic regression (LR), neural networks (ANN) and random forest (RF). We select LR for its popularity in credit approval applications; we select ANN for it being the best-performing individual classifier on some credit approval datasets [69]; we select RF for it being the best-performing homogeneous ensemble on some credit approval datasets [69], as HC can be viewed as a homogeneous ensemble method. For the sake of exposition, we present numbers just for baselines based on LR, as they perform relatively better.

Second, for the purposes of comparison, we include MAJ – predict the *majority* label if examples with the exact same feature values appeared in the training set, and reject if not – which can be thought of as a non-strategic counterpart of MINCUT. We also include k-nearest neighbors (KNN) as a baseline, since it is closely related to MAJ.

These out-of-the-box classifiers need help dealing with missing data, whether they are missing naturally at training and test time or strategically at test time, and to this end, we employ (a) IMP: mean/mode imputation [69], and (b) R-F: reduced-feature modeling [89], for each of them.

When the dataset has a large number of features, MINCUT and IC-LR can be directly applied. For HC, we assist it in two ways: (a) by selecting 4 features based on the training data, denoted by FS (feature selection),[11] and (b) by choosing a limited number of small subsets (30 with 1 feature and 30 with 2 features), denoted by APP (approximation). Note that since our proposed methods are truthful, we can assume that features are reported as is. However, for all out-of-the-box classifiers, except IMP(LR), it is infeasible to simulate strategic withholding of feature values, due to the enormous number of combinations of features.

Last but not least, we test all methods with the discretization of continuous features (into categorical ones) [42], for reasons given in earlier sections.

[11]Such a technique can be applied to other methods too – the results (see the Supplement of [66]) are not very different from those in Tables 4.4.

| Classifier | Australia | | Germany | | Poland | | Taiwan | |
|---|---|---|---|---|---|---|---|---|
| | Tru. | Str. | Tru. | Str. | Tru. | Str. | Tru. | Str. |
| HC(LR) | .792 | **.792** | .639 | .639 | .659 | .659 | .648 | .648 |
| MINCUT | .770 | .770 | .580 | .580 | .501 | .501 | .652 | **.652** |
| IC-LR | .788 | .788 | .654 | **.654** | .639 | .639 | .499 | .499 |
| IMP(LR) | .796 | .791 | **.663** | .580 | **.714** | **.660** | **.670** | .618 |
| R-F(LR) | **.808** | .545 | .631 | .508 | .670 | .511 | .665 | .590 |

Table 4.3: Our methods vs. the rest: mean classifier accuracy for $\epsilon = 0.2$, balanced datasets, 4 features

| Classifier | Australia | | Germany | | Poland | | Taiwan | |
|---|---|---|---|---|---|---|---|---|
| | Tru. | Str. | Tru. | Str. | Tru. | Str. | Tru. | Str. |
| HC(LR) w/ disc. | .794 | .794 | .641 | .641 | .692 | .692 | .650 | **.650** |
| MINCUT w/ disc. | .789 | .789 | .629 | .629 | .692 | .692 | .649 | .649 |
| IC-LR w/ disc. | **.800** | **.800** | .651 | **.651** | .698 | **.698** | .646 | .646 |
| IMP(LR) w/ disc. | .799 | .762 | **.652** | .577 | **.719** | .631 | **.686** | .541 |
| R-F(LR) w/ disc. | .796 | .542 | .633 | .516 | .708 | .522 | .684 | .587 |

Table 4.4: Our methods vs. the rest: mean classifier accuracy for $\epsilon = 0.2$, balanced datasets, 4 features ("w/ disc." stands for "with discretization of features")

**Results** Here, we report results only for $\epsilon = 0.2$. We also limit our exposition of HC, IMP and R-F methods to those based on logistic regression, as these perform better than their ANN/RF/KNN counterparts. For a comprehensive compilation of all results, along with standard deviation numbers, please refer to the Supplement of [66].

*With a small number of features (Table 4.3):* As expected, the out-of-the-box baselines perform well under truthful reporting, but not with strategic reporting. Our methods are robust to strategic withholding, and in line with the earlier discussion on the potential issues faced by MINCUT and IC-LR (in Sections 4.3 and 4.5), we see that **(a)** HC(LR) performs most consistently, and **(b)** in some cases, MINCUT (e.g., Poland) and IC-LR (e.g., Taiwan) do not do well.

*With discretization (Table 4.4):* As expected, discretization of numerical features into binary categories improves the performance of MINCUT and IC-LR, for reasons explained in Sections 4.3 and 4.5 respectively. We also see some benefit from discretization for HC(LR) when the features are mostly continuous (e.g., Poland), and less so when they are already discrete (e.g., Taiwan).

*With a large number of features (Table 4.5):* We see broadly similar trends here, except that in the case with discretization, IC-LR performs much better than before (e.g., Poland). The reason for this is that IC-LR is able to use all the available features once they are discretized into binary categories. However, without discretization, HC methods are more reliable (e.g., Poland and Taiwan).

*On the out-of-the-box baselines:*

- *Imputation-based methods* are sensitive vis-á-vis the mean/mode values used. There is incentive to drop a certain feature if the imputed value is a positive signal. If there are many such features, then these methods perform poorly, as seen in Table 4.5 (cf. Table 4.3, Australia). If the imputed values do not give a clear signal (e.g., when the distribution of each feature value is not skewed), there is a high variance in the performance of these methods

| Classifier | Australia | | Germany | | Poland | | Taiwan | |
|---|---|---|---|---|---|---|---|---|
| | Tru. | Str. | Tru. | Str. | Tru. | Str. | Tru. | Str. |
| HCFS(LR) | .795 | .795 | .625 | .625 | .678 | .678 | .648 | .648 |
| HCAPP(LR) | .777 | .777 | .617 | .617 | .658 | .658 | .638 | .638 |
| MINCUT | .496 | .496 | .499 | .499 | .499 | .499 | .499 | .499 |
| IC-LR | .798 | .798 | .654 | **.654** | .607 | .607 | .588 | .588 |
| HCFS(LR) w/ disc. | .794 | .794 | .632 | .632 | .694 | .694 | .649 | .649 |
| HCAPP(LR) w/ disc. | .782 | .782 | .620 | .620 | .724 | .724 | .644 | .644 |
| MINCUT w/ disc. | .534 | .534 | .503 | .503 | .499 | .499 | .550 | .550 |
| IC-LR w/ disc. | .805 | **.805** | .653 | .653 | .773 | **.773** | .667 | **.667** |
| IMP(LR) | .802 | .701 | **.663** | .523 | .729 | .507 | .657 | .501 |
| IMP(LR) w/ disc. | **.809** | .723 | .659 | .554 | **.783** | .503 | **.697** | .501 |

Table 4.5: Our methods vs. the rest: mean classifier accuracy for $\epsilon = 0.2$, balanced datasets, all features

(see the Supplement of [66]). In some cases, the benchmarks perform as well as, or slightly better than, our incentive-compatible classifiers. For example, in Table 4.3, for the Australia and Poland data sets, the accuracy of IMP(LR) and that of HC(LR) are within $0.001$ of each other. This happens because the imputed values are, in these cases (but not in most of our other cases), negative indicators of the positive label, and therefore there is generally no incentive to strategically drop features.

- *Reduced-Feature modeling*, despite performing well under truthful reporting, allows too many examples to be accepted under strategic reporting, which hurts its performance. This is true especially for smaller $\epsilon$, as each subclassifier has fewer examples to train on, giving several viable options for strategic withholding.

We note here that the variance (in the accuracy achieved) produced by our methods, since they are robust to strategic withholding, is much smaller than that of the baseline methods (exact numbers can be found in the Supplement of [66]).

## 4.7 Conclusion

In this chapter, we studied the problem of classification when each agent at prediction time can strategically withhold some of its features to obtain a more favorable outcome. We devised classification methods (MINCUT, HC and IC-LR) that are robust to this behavior, and in addition, characterized the space of all possible truthful classifiers in our setting. We tested our methods on real-world data sets, showing that they outperform out-of-the-box methods that do not account for the aforementioned strategic behavior.

An immediate question that follows is relaxing the assumption of having access to truthful training data – for example, one could ask what the best incentive-compatible classifier is given that the training data consists of best responses to a known classifier $f$; or, one could consider an online learning model where the goal is to bound the overall loss over time. A much broader question for future work is to develop a more general theory of robustness to missing data that naturally includes the case of strategic withholding.

# Chapter 5

# When Samples Are Strategically Selected

## 5.1 Introduction

In previous chapters, we have investigated various aspects of classification in the presence of strategic behavior. Now let us deviate from this classical model of classification, and consider a sample-based setting, where each data point is associated with multiple samples, each of which provides some information about the true qualifications of the data point. In such classification problems, the common assumption is that we have access to *all* the samples. However, in many contexts, we either do not have direct access to the samples, or we can inspect only a limited set of samples and do not know which are the most relevant ones. In such cases, we must rely on another party (the *agent*) to either provide the samples or point out the most relevant ones. This agent may have different incentives, which raises several concerns. One is that the individual samples cannot even be trusted—e.g., we ask for images but the agent manipulates the images with editing software before sending them. This is an issue that we do not consider in this chapter (in fact, this will be the main focus of the next chapter, Chapter 6); we assume that the agent cannot modify samples or create fake samples. (In the related research discussion below, we list some work that does not make this assumption.) But even in a context where the individual samples can be trusted (e.g., using cryptographic tools like digital signatures [45, 49]), there is still the concern that the agent sends only a *biased collection* of samples. If we do not know how many samples $n$ the agent has available, then we cannot know whether the agent has submitted all of them. Moreover, we may have capacity to deal with only a fixed number $m$ of samples.

Consider the following scenario. A faculty member (the agent) wishes to convince the chair of the department (the *principal*) to interview a particular candidate, while the chair wants to interview the strongest candidates.[1] The principal has time to read exactly three of the candidate's papers, and asks the agent, who is familiar with all of the candidate's work, which three she should read. Naturally, the agent chooses the best three papers. The principal, when reading them, then knows that these three papers may not be representative of all the candidate's work, and should make her decision with this in mind.

[1] Of course the decision may not rest with the chair (alone); if so, we may think of the body of people that needs to be convinced as the principal.

In fact, it is not clear that the principal should just ask for the three best papers according to a single metric. For example, she could also say: "Of the three papers, at least one must have at most two authors." Or: "At least one of the papers must introduce a new problem." Or: "In at least one of the papers, the 42nd letter must be a vowel." They also do not need to be hard constraints; she could just announce that she would appreciate it if one of the papers has at most two authors, but if that is not the case and the three papers are spectacular she would still interview the candidate. In general, the principal will set a *policy* for when she would interview a candidate, and the agent will base his choice of papers on this policy.[2]

Similar examples abound where an agent aims to convince a principal based on limited data. The above example can be generalized to hiring in many other professions: for example, in sports, high school coaches may want to convince a college scout to come out and take a look at a player. As before, the scout (the principal) has limited time, and she must make a decision of taking the trip or not, based on the limited video footage of that player provided by the coach (the agent). The scout can specify a policy and the coach will choose the footage accordingly. For example, the scout may just wish to see the best moments, or have a policy that she prefers to see footage of the player in multiple roles (offense, defense, ...). The decisions do not need to be hiring decisions, of course. For example, a city may make a bid to host an event based (in part) on a few selected photographs of the surroundings. The committee deciding on the location will probably appreciate having photographs of both the venue and of the nearby beach, rather than multiple photographs of just one of these.

### 5.1.1 Our Results

In this section, we highlight some of our structural and computational results.

We study the problem of designing an optimal classification policy when the samples are provided by a strategic agent. The agent has $n$ samples, and while she cannot modify the samples, she can choose which $m$ samples to submit. Our work is motivated by questions such as the following:

1. How does strategic sample selection affect the principal's classification problem? Is it easier or harder compared to when she has direct access to $m$ samples?
2. Are there conditions under which she should just ask for the best $m$ samples according to a single metric? Are there conditions under which this is not optimal?
3. What is an optimal policy for the principal?

To answer question (1), we give two examples (Examples 5.1 and 5.2) that show that, depending on the instance, classification based on strategically selected samples could be easier or harder for the principal compared to when she has direct access to the samples.

For question (2), we prove several structural results. We show that when a single sample is reported ($m = 1$) or when all samples are reported ($m = n$), any optimal policy should accept reports whose good vs. bad likelihood ratio (i.e., a single metric) exceeds a certain threshold (Theorems 5.1 and 5.2). However, such a characterization fails to hold for the general $1 < m < n$ case (Proposition 5.3).

[2]Note that the candidate is *not* a strategic player; the candidate takes no actions in the game.

For question (3), for the case of one good and one bad distribution, we show that for any value of $m$, we can design a policy that has good behavior in the limit (Corollary 5.2 and Theorem 5.3). When there are multiple good and bad distributions, we show that if $m = 1$ and the distributions are piecewise constant, we can efficiently compute policies with target accepting probabilities (Theorem 5.6).

## 5.2 Preliminaries

The principal is interested in the underlying *type* $\theta \in \Theta$ of the entity about which she is making the decision. For example, in much of the chapter we will consider a binary type space $\Theta = \{g, b\}$ ("good" and "bad"). The entity generates $n$ *samples* from a sample space $X$,[3] where samples $x$ are drawn i.i.d. (so with replacement) according to $P(x|\theta)$. We sometimes use the shorthand $\theta(x) = P(x|\theta)$. These $n$ samples constitute a multiset[4] $D$ which is the *dataset* available to the agent. The agent must select a multiset $R \subseteq D$ [5] with $|R| = m$ as his *report* to the principal. The principal must decide to accept or reject, based on the report. She commits to a policy. A *randomized policy* $\Pi_m : \mathcal{R}_m \to [0, 1]$ assigns to each report a probability of acceptance. Here $\mathcal{R}_m$ is the set of all possible reports of size $m$. A *deterministic policy* has $\Pi_m(R) \in \{0, 1\}$ for all $R$; it is equivalently defined by the set of accepted reports, $\mathcal{S}_m = \{R : \Pi_m(R) = 1\}$.

The agent aims to maximize the probability of acceptance regardless of the true type. He will choose to report some $R^\star$ in $\operatorname{argmax}_{R \subseteq D} \Pi_m(R)$. The principal, taking into account the strategic behavior by the agent, chooses her policy $\Pi_m$ to attain her own objectives. For example, she may have a *utility* $u(\theta)$ for accepting $\theta$; when $\Theta = \{g, b\}$, $u(b) < 0 < u(g)$. When combined with a prior over $\Theta$, this creates a well-defined optimization problem for the principal. Alternatively, she may have *target acceptance probabilities* for each type; for example, she may want to accept $b$ with probability at most $p_b$ and $g$ with probability at least $p_g$ (corresponding to limits on the two types of errors). This does not require a prior over $\Theta$.

For simplicity, we use the following notation in proofs interchangeably. For *one-sample* policies ($m = 1$), we sometimes denote the policy by its accepted set of samples $S$ (instead of $\mathcal{S}$, which is a family of singleton multisets). That is, $S = \{x \mid \{x\} \in \mathcal{S}\}$. For a type, e.g., $g \in \Theta$, we abuse notation and use $g(S)$ to denote the total probability mass of $g$ on a set $S \subseteq X$. That is, $g(S) = \Pr_{x \sim g}[x \in S]$.

### 5.2.1 Illustrative Examples

We now present a few examples that illustrate some of the key issues. The first example illustrates that the strategic selection sometimes helps the principal. In the language of our motivating

[3]For simplicity, we assume the sample space $X$ is in some Euclidean space, i.e., $X \subseteq \mathbb{R}^d$ for some integer $d > 0$.

[4]Recall that a multiset is a set in which an element may occur more than once. One could also think of this as a vector, but the order of the elements does not matter in our context.

[5]We use "$\subseteq$" for the standard subset notion on multisets. For multisets $A$ and $B$, $A \subseteq B$ iff $c_B(x) \leq c_A(x)$ for all $x \in B$, where $c_S(x)$ is the number of occurrences of $x$ in $S$.

example, this is the case where the principal and the agent can only classify papers as high- or low-quality, and high-quality papers are rare.

**Example 5.1.** Let $\Theta = \{g, b\}$ and $X = \{0, 1\}$. Let $g(1) = 0.05$ and $b(1) = 0.005$. Let $n = 50$ and $m = 1$. The natural policy is to accept iff the agent submits report $\{1\}$. In other words, the agent can convince the principal to accept iff at least one of the $n$ papers has high-quality. The probability a good type is accepted is $1 - 0.95^{50} \approx 0.92$; for a bad type it is $1 - 0.995^{50} \approx 0.22$. In this example, thanks to the agent's strategic selection, the principal can classify quite effectively while only observing a single sample; in contrast, if she had to observe samples directly, a single sample would give her very little information.

However, the next example (where high-quality papers are less rare) shows that the opposite can also happen. Strategic selection can make it much harder for the principal to distinguish between good and bad types.

**Example 5.2.** Let $\Theta = \{g, b\}$ and $X = \{0, 1\}$. Let $g(1) = 0.95$ and $b(1) = 0.05$. Let $n = 50$ and $m = 1$. Again, the natural policy is to accept iff the agent submits $\{1\}$. The probability that a good type is accepted is $1 - 0.05^{50} \approx 1$; for a bad type it is $1 - 0.95^{50} \approx 0.92$. In this example, the strategic selection of samples by the agent makes it very difficult for the principal to distinguish between $g$ and $b$; in contrast, if she could observe samples directly, a single sample would allow her to classify quite effectively.

Fortunately, there's a workaround: the principal can effectively reduce $n$ by specifying an irrelevant (i.e., uncorrelated with the type) requirement, such as that the 42nd letter of the paper should be a vowel.[6] Since there will generally be nearly infinitely many irrelevant attributes of the samples, we assume that each sample is associated with a real number drawn uniformly[7] from $(0, 1)$, representing the irrelevant information.

**Example 5.3.** Let $\Theta = \{g, b\}$ and $X = \{0, 1\} \times (0, 1)$ (where the first number represents the relevant information and the second number the irrelevant information). Let $g(\{1\} \times (0, 1)) = 0.95$ and $b(\{1\} \times (0, 1)) = 0.05$. Note that this is the same example as Example 5.2, except for the additional irrelevant information. Let $n = 50$ and $m = 1$. Now consider the policy that accepts $\mathcal{S} = \{\{x\} : x \in \{1\} \times (0, 0.05)\}$. That is, the principal accepts only samples with good relevant information and irrelevant information that has a 1 in 20 chance of occurring. The probability that a good type is accepted is $1-(1-0.95 \cdot 0.05)^{50} \approx 0.92$; for a bad type it is $1-(1-0.05 \cdot 0.05)^{50} \approx 0.12$. Thus, the addition of irrelevant information allows the principal to classify much more effectively.

Example 5.3 illustrates that the difficulty of Example 5.2 is primarily due to the discreteness of the sample space.

Our last example shows that with multiple bad distributions, the optimal policy does not evaluate the quality of samples individually, but rather considers them in combination. One of the questions of interest later in this chapter is under which circumstances this can happen.

**Example 5.4.** Let $\Theta = \{g, b_1, b_2\}$ and $X = \{0, 1\}$. Let $g(1) = 0.5$, $b_1(1) = 0.99$, $b_2(1) = 0.01$.

[6] One may wonder whether this creates an incentive for candidates to write their papers in a particular way; this can be avoided by choosing this requirement close to the decision. In any case, we do not consider the process that originally produces the $n$ samples as a strategic entity in this chapter.

[7] Any continuous distribution can be transformed to a uniform one, using the standard trick of applying the CDF to the drawn number first.

Let $n = 10$ and $m = 2$. Consider the policy that accepts $\mathcal{S} = \{\{0, 1\}\}$, i.e., it accepts if both possible samples are reported. The probability that a good type is accepted is $1 - 2 \cdot 0.5^{10} \approx 0.998$; the probability that a bad type is accepted is less than $1 - 0.99^{10} \approx 0.10$. In contrast, if we accept reports that have the same sample twice, then one of the two bad types is extremely likely to succeed.

## 5.3 Basic Results

In this section, we provide some basic results that justify why we focus on deterministic policies and continuous distributions in the rest of the chapter.

### 5.3.1 Deterministic vs. Randomized Policies

In this subsection, we discuss the relative power of deterministic and randomized policies, justifying our focus on deterministic policies in the rest of the chapter.

We first show that in our setting, randomized policies can be decomposed into a distribution over deterministic policies.

**Proposition 5.1.** *Any randomized policy can be decomposed into a distribution over deterministic policies, such that the accepting probability of any report remains the same.*

As a simple corollary, if the principal is trying to maximize her utility, one of the deterministic policies must be optimal.

**Corollary 5.1.** *Assume there is a prior over $\Theta$ and the principal has a utility $u(\theta)$ for accepting type $\theta \in \Theta$. If the principal wishes to maximize her expected utility, there exists a deterministic policy that is optimal.*

We defer the proofs of Proposition 5.1 and Corollary 5.1 to the appendix.

It should be noted that Corollary 5.1 is generally not true in mechanism design. For example, for designing revenue-maximizing auctions, it is well-known that the optimal mechanism may require randomization [54]. In fact, even in our problem, if the principal has target acceptance probabilities for each type, for example "I want to accept at least 90% of good candidates and at most 10% of bad candidates," then it is possible that only randomized policies obtain these goals simultaneously, as the following example demonstrates.

**Example 5.5.** Let $\Theta = \{g, b\}$ and $X = \{0, 1\}$. Let $g(0) = 1/2$, $g(1) = 1/2$, $b(0) = 1$, $b(1) = 0$, and $n = m = 1$. If we wish to accept the good distribution at least $3/4$ of the time, and accept the bad distribution at most $1/2$ of the time, we have to use a randomized policy: always accept $\{1\}$ and accept $\{0\}$ with probability $1/2$.

On the other hand, this example heavily relies on the discrete nature of the sample space; if we make the sample space continuous—say, $\{0, 1\} \times (0, 1)$ where the second number is drawn uniformly at random, as in Example 5.3—then we can achieve the same result with a deterministic mechanism, by effectively using the second number to generate the randomness.

In practice, deterministic policies have several other advantages as well. They are straightforward to implement, transparent, fair, and not subject to willful manipulation of the random numbers. For all these reasons, we focus on deterministic policies in the rest of this chapter.

### 5.3.2 Continuous vs. Discrete Distributions

In this subsection, we compare continuous and discrete distributions, justifying our focus on continuous distributions in the rest of the chapter. As we will prove in the next proposition, given our focus on deterministic policies, for discrete distributions we face NP-hardness, simply due to having to solve a knapsack problem. We defer the proof of Proposition 5.2 to the appendix.

**Proposition 5.2.** *Given two discrete distributions $g$ and $b$, and two target acceptance probabilities $p_g$ and $p_b$, it is NP-hard to decide if there exists a deterministic policy that accepts $g$ with probability at least $p_g$ and accepts $b$ with probability at most $p_b$. This is true even when $n = m = 1$.*

In contrast, Theorem 5.6 states that for continuous distributions, we can solve the decision problem of Proposition 5.2 efficiently in much more general settings.

As we discussed in Example 5.3, in practice, we can often make the distribution over the sample space effectively continuous, by considering irrelevant information. For example, when considering video footage of an athlete, we may be able to summarize all the relevant information in discrete terms (did the athlete make the shot or not, etc.). Meanwhile, the background of the video provides almost unlimited irrelevant information (is someone eating popcorn in the background, etc.). Imagine that we can only watch a single video clip of a basketball player. If our policy is to only accept if the player (say) makes a 3-pointer, then even mediocre players will be able to produce such a clip. But if our policy is to only accept if the player makes a 3-pointer while a girl is eating popcorn in the background, then only players who frequently score 3-pointers are likely to be able to produce such a clip. From a technical viewpoint, we can assume that each sample is associated with a random real number drawn uniformly from $(0, 1)$, as in Example 5.3.

For all these reasons, we focus on continuous distributions in the rest of this chapter. We now move on to study strategic sample selection in these more restricted settings.

## 5.4 One Good and One Bad Distribution

In this section, we consider the setting in which the type space is binary: $\Theta = \{g, b\}$.

### 5.4.1 One Sample

We first consider the special case where the agent submits only one sample ($m = 1$). Our main structural result (Theorem 5.1) states that any Pareto optimal policy takes the following form: accept all reports $\{x\}$ such that the *likelihood ratio* $g(x)/b(x)$ is greater than some threshold.

As a corollary of Theorem 5.1, when the agent has sufficiently many samples ($m = 1$ and $n \to \infty$), we can characterize the optimal tradeoff between the accepting probabilities $p_g$ and $p_b$ (Corollary 5.2). This tradeoff is quantitatively governed by the maximum likelihood ratio $\max \frac{g(x)}{b(x)}$ over the entire sample space $x \in X$. This is in contrast to the distribution learning literature [10, 17, 84], where this tradeoff is often determined by some global measure (e.g., total variation distance) between the two distributions.

Let $\Pi$, $\Pi'$ be two policies that accept the good distribution with probability $p_g$ and $p'_g$, and accept the bad distribution with probability $p_b$ and $p'_b$ respectively. We say $\Pi'$ is *strictly better* than

$\Pi$ if $p'_g \geq p_g$ and $p'_b \leq p_b$ and at least one of the two inequalities holds strictly. We say $\Pi$ is *Pareto optimal* if there is no other policy $\Pi'$ strictly better than $\Pi$.

**Theorem 5.1.** *Suppose $m = 1$ and we have continuous distributions $g$ and $b$. Consider any optimal deterministic policy $\Pi$. Let $S \subseteq X$ be the accepting region of $\Pi$. Then, for any point $x_1$ strictly inside $S$ and any $x_2$ strictly outside of $S$,*

$$g(x_1)/b(x_1) \geq g(x_2)/b(x_2).$$

Before proving Theorem 5.1, we first discuss its conditions and implications. We can always change the policy on a set of measure zero, and the resulting policy is equivalent to the original one. Therefore, the condition in Theorem 5.1 only applies to the interior of $S$ and $X \setminus S$. An immediate consequence of Theorem 5.1 is that, if the principal's utility only depends on the accepting probabilities, and is strictly monotonically increasing in $p_g$ and strictly monotonically decreasing in $p_b$, then the optimal policy must be Pareto optimal and hence it must satisfy the condition in Theorem 5.1.

*Proof.* Let $p_g$, $p_b$ be the probabilities that $\Pi$ accepts the good and bad distributions respectively.

$$p_g = \Pr_{D \sim g^n}[D \cap S \neq \emptyset] = 1 - (1 - g(S))^n,$$
$$p_b = \Pr_{D \sim b^n}[D \cap S \neq \emptyset] = 1 - (1 - b(S))^n.$$

Suppose there is $x_1$ strictly in $S$ and $x_2$ strictly outside $S$, where

$$g(x_1)/b(x_1) < g(x_2)/b(x_2).$$

Pick neighborhoods $N_1$ and $N_2$ of $x_1$ and $x_2$, in $S$ and $X \setminus S$ respectively, such that $g(N_1) = g(N_2) > 0$ and $g(N_1)/b(N_1) < g(N_2)/b(N_2)$. Such neighborhoods exist because the likelihood ratio $g(x)/b(x)$ is a continuous function, and both $x_1$ and $x_2$ are not on the boundary of $S$.

We will show that a different policy $\Pi'$ with accepting region $S' = (S \setminus N_1) \cup N_2$ is a better policy. Let $p'_g$ and $p'_b$ be the accepting probabilities of $\Pi'$. Since $g(S') = g(S)$ and $b(S') = b(S) - b(N_1) + b(N_2) < b(S)$, we have

$$p'_g = 1 - (1 - g(S'))^n = 1 - (1 - g(S))^n = p_g, \text{ and}$$
$$p'_b = 1 - (1 - b(S'))^n < 1 - (1 - b(S))^n = p_b. \qquad \square$$

**Corollary 5.2.** *Fix continuous distributions $g$ and $b$. Let $r = \sup_{x \in X}(g(x)/b(x))$ be the maximum likelihood ratio over the entire sample space $X$.[8] When $m = 1$ and $n \to \infty$, any Pareto optimal deterministic policy $\Pi$ satisfies*

$$p_g + (1 - p_b)^r = 1,$$

*where $p_g$ and $p_b$ are the probabilities that $\Pi$ accepts $g$ and $b$ respectively.*

[8]For simplicity, we assume the maximum likelihood ratio $r = \sup_x(g(x)/b(x))$ exists and is finite. A similar argument shows that when $r = \infty$, we can get policies with $p_g = 1$ and $p_b = 0$.

*Proof.* Fix any $0 < \varepsilon < 1$. Because the likelihood ratio function $g(x)/b(x)$ is continuous and its supremum is $r$, there exists a small neighborhood $S \subseteq X$ such that $g(x)/b(x) \geq (1-\varepsilon)r$ for all $x \in S$. Recall that $g(S) = \Pr_{x\sim g}[x \in \mathcal{N}] > 0$ is the probability that a random sample from the good distribution $x \sim g$ is in $S$. By the definition of $S$, we know that $b(S) \leq g(S)/(r(1-\varepsilon))$.

Consider the policy that accepts iff the agent reports a sample in $S$. We will show that $S$ is essentially optimal as $n \to \infty$. Let $p_g$ and $p_b$ denote the accepting probabilities of $S$. For notational convenience let $\delta = g(S)$. We have

$$p_g = (1 - (1 - g(S))^n) = 1 - (1-\delta)^n, \text{ and}$$
$$p_b = (1 - (1 - b(S))^n) \leq 1 - \left(1 - \frac{\delta}{r(1-\varepsilon)}\right)^n.$$

We can rewrite the inequality on $p_b$ by substituting $\delta$. Because the inequality holds for any $\varepsilon$ and $n$, we can let $\varepsilon \to 0$ and $n \to \infty$ and get

$$p_b \leq 1 - \left(1 - \frac{1-(1-p_g)^{1/n}}{r(1-\varepsilon)}\right)^n \to 1 - (1-p_g)^{1/r}.$$

On the other hand, the upper bound on $p_b$ is tight when $\varepsilon = 0$. This is because the acceptance region $S'$ of any deterministic policy can have likelihood ratio at most $r$, and thus $b(S') \geq g(S')/r$ and a similar calculation gives the same lower bound on $p_b$. Therefore, we can conclude that for any Pareto optimal policy, $p_g + (1-p_b)^r = 1$ as $n \to \infty$. □

### 5.4.2 Multiple Samples

We now move on to the case where the agent submits $m$ samples ($m > 1$). We first generalize the notion of the likelihood ratio to reports of multiple samples.

**Definition 5.1.** We define the likelihood ratio of a report $R$ to be the product of the samples' likelihood ratios $\prod_{x\in R} \frac{g(x)}{b(x)}$, as if the samples in $R$ are drawn i.i.d. from the distribution.

The following theorem states that when $m = n$, any Pareto optimal policy essentially accepts reports whose likelihood ratio exceeds some threshold.

**Theorem 5.2.** *Suppose $m = n$ and we have continuous distributions $g$ and $b$. Consider any Pareto optimal deterministic policy $\Pi$ which accepts all and only reports in $\mathcal{S}$. Then, for any report $R_1$ strictly inside $\mathcal{S}$ and any $R_2$ strictly outside of $\mathcal{S}$, we have*

$$\prod_{x\in R_1} g(x)/b(x) \geq \prod_{x\in R_2} g(x)/b(x).$$

We defer the proof of Theorem 5.2 to the appendix.

Note that for the case $m = 1$, Theorem 5.1 states that in that case, too, the agent should report the sample that maximizes the likelihood ratio. Thus, it is natural to conjecture that this continues to hold when $m < n$. This, however, is false. In fact, we can show that the optimal policy does not admit even the following weaker structural property.

**Definition 5.2.** A policy *orders the sample space* if there exists an ordering on the elements of the sample space, such that an optimal response for the agent is to always report his highest samples in this ordering.

In the $m = 1$ and $m = n$ cases, the optimal policy based on the likelihood ratio clearly satisfies this property, ordering the sample space by likelihood ratio. (In the $m = n$ case, this is because the likelihood ratio is the product of the likelihood ratios of the individual samples, and this product is maximized by choosing the samples that maximize that ratio.) The following proposition shows that this does not hold in general for $1 < m < n$.

**Proposition 5.3.** *When $1 < m < n$, sometimes a Pareto optimal policy does not order the sample space.*

*Proof.* For simplicity, we present a counter example that is a discrete distribution. It can be easily changed to a continuous distribution without affecting any part of the argument.

Let $\Theta = \{g, b\}$, $X = \{0, 1, 2\}$, and

$$\begin{aligned} g(0) &= 0, \quad & g(1) &= 0.1, \quad & g(2) &= 0.9, \quad & \text{and} \\ b(0) &= 0.8, \quad & b(1) &= 0.1, \quad & b(2) &= 0.1. \end{aligned}$$

Let $m = 2$ and $n = 3$, i.e., the agents has 3 i.i.d. samples and chooses 2 of them to submit.

We claim a policy $\Pi$ that accepts reports $\{1, 1\}$ and $\{2, 2\}$ is Pareto optimal. First notice that $\Pi$ accepts $g$ with probability 1. Since an agent does not draw $\{0\}$ from $g$, so by the pigeonhole principle, among the $n = 3$ samples there must be either two copies of $\{1\}$ or two copies of $\{2\}$. On the other hand, the principal must accept these two reports with probability 1 if she wants to always accept $g$. This is because when $\theta = g$, the agent's data $D$ could be $\{1, 1, 1\}$ (or $\{2, 2, 2\}$), in which case he is forced to report $R = \{1, 1\}$ (or resp. $\{2, 2\}$). Therefore, among all policies, $\Pi$ has the smallest probability of accepting $b$.

The above example rules out structural results in the form of Theorems 5.1 and 5.2, because the report $\{1, 2\}$ has higher likelihood ratio than $\{1, 1\}$. Furthermore, note that any policy that accepts $\{1, 1\}$ and $\{2, 2\}$ but not $\{1, 2\}$ cannot order the sample space; for example, if 1 were ordered at least as high as 2, then an agent with data $\{1, 2, 2\}$ can report $\{1, 2\}$ instead of $\{2, 2\}$ according to the ordering, but this is suboptimal because $\{1, 2\}$ is rejected while $\{2, 2\}$ is accepted. □

It of course remains possible that there is an elegant way to describe the optimal policy in this context, but Proposition 5.3 rules out many natural possibilities. However, if we are willing to give up on exact optimality, then we can still define a policy that performs reasonably well in the limit. Theorem 5.3 gives a policy whose error probability (probability of rejecting $g$ or accepting $b$) decreases exponentially in $m$. This error guarantee is similar to the setting where the principal has direct access to $m$ samples, in which case the failing probability also decreases exponentially in $m$. The difference is that, as in Corollary 5.2, the coefficient in the exponent depends on the maximum likelihood ratio $r$ rather than (say) the total variation distance between $g$ and $b$.

**Theorem 5.3.** *Fix $m \geq 1$ and two continuous distributions $g$ and $b$. Let $r = \sup_x \frac{g(x)}{b(x)} > 1$ denote the maximum likelihood ratio. As $n \to \infty$, there is a deterministic policy whose error probability is at most* $\exp\left(-\frac{1}{2}(1 - r^{-1/2})^2 m\right)$.

The policy that achieves Theorem 5.3 focuses on a small region $S$ where $g(S) \gg m/n$ and $b(S) \ll m/n$. This way, for $n$ samples drawn from $g$, in expectation, $ng(S) \gg m$ samples are from $S$; and for $n$ samples drawn from $b$, $nb(S) \ll m$ samples are from $S$. Therefore, if we accept all reports with $m$ samples from $S$, we can distinguish $g$ from $b$. We defer the proof of Theorem 5.3 to the appendix.

## 5.5 One Good and Multiple Bad Distributions

*For men are good in but one way, but bad in many.*

— Aristotle, *Nicomachean Ethics*

We now consider the case where there are multiple bad distributions, but still only a single good one.

### 5.5.1 One Sample

Again, we first investigate the single-sample case ($m = 1$). We first show there are cases in which no policy can perform well across all possible priors over $\Theta$.

**Example 5.6.** Let $\Theta = \{g, b_1, b_2\}$ and $X = \{0, 1\} \times (0, 1)$. Let $g(\{0\} \times (0, 1)) = g(\{1\} \times (0, 1)) = 0.5$, $b_1(\{0\} \times (0, 1)) = 1$, and $b_2(\{1\} \times (0, 1)) = 1$. There is no policy which makes the right decision with probability larger than $0.5$ against any prior. This is because any deterministic policy has the following form: for some $p, q \in [0, 1]$, the policy accepts all reports in $\mathcal{S} = \{\{x\} \mid x \in (\{0\} \times (0, p)) \cup (\{1\} \times (0, q))\}$. To accept $g$ w.p. larger than $1/2$, we need $p + q > 1$. W.l.o.g. assume $p > 1/2$, but then the policy accepts $b_1$ w.p. $p > 1/2$.

When there is a prior over $\Theta$ and the principal has utilities for accepting each type $\theta \in \Theta$, the next theorem characterizes the behavior of optimal policies in the limit. Note that Theorem 5.4 does not hold if we have specific target acceptance probabilities for individual bad distributions.

**Theorem 5.4.** *Fix $m = 1$ and a partition of the sample space $X$ into $t$ pieces. Let $\Theta = \{g, b_1, \ldots, b_k\}$. Assume every distribution is constant on every piece, the principal has utility $u(\theta)$ for accepting type $\theta$, and there is a prior $q$ over $\Theta$. Then, for sufficiently large $n$, there is a utility-maximizing policy that accepts only reports in a subset of one single piece (modulo accepting any report that has measure zero).*

*Proof.* Suppose the optimal policy accepts all reports in $S$ and $S$ overlaps with multiple pieces $P_1, \ldots, P_t$. Let $(g(S_j), b_1(S_j), \ldots, b_k(S_j))$ denote the "cumulative" probabilities of the types in $S_j = S \cap P_j$. Let $\alpha_i^\theta$ be the density of distribution $\theta$ on $P_i$, and let $\beta_{ij} = b_i(S_j)/g(S_j) = \alpha_j^{b_i}/\alpha_j^g$ be the likelihood ratio $b_i(x)/g(x)$ on piece $j$. We argue that moving all the mass, measured by $g$, to one of the $k$ pieces achieves at least the same probability of success. Since $n$ is large enough,

$\sum_j g(S_j)$ can be contained in any of the $t$ pieces. The expected utility of $S$ is

$$q_g u(g)\left(1-\left(1-\sum_j g(S_j)\right)^n\right) + \sum_i q_{b_i} u(b_i)\left(1-\left(1-\sum_j \beta_{ij} g(S_j)\right)^n\right).$$

Let $\beta_i = (\beta_{i1}, \ldots, \beta_{i\ell})$, $\gamma = (g(S_1), \ldots, g(S_t))$. The principal's expected utility can be written as

$$u(g)q_g(1-(1-\|\gamma\|_1)^n) + \sum_i u(b_i)q_{b_i}(1-(1-\beta_i^\top\gamma)^n).$$

Note that since $0 \le \beta_i^\top\gamma \le 1$, $(1-\beta_i^\top\gamma)^n$ is convex in $\gamma$ for any $i$. Fixing $\|\gamma\|_1$, Since $q_{b_i} \ge 0$ and $u(b_i) < 0$, the overall utility is also convex in $\gamma$. Therefore, the maximum utility is achieved when $\gamma$ has only one non-zero entry. Equivalently, the optimal policy should focus on a single piece. □

Theorem 5.4 crucially relies on there being only a single good distribution, as the following example demonstrates.

**Example 5.7** (non-locality with multiple good distributions)**.** Let $\Theta = \{g_1, g_2, b\}$ and $X = \{0,1,2\} \times (0,1)$. Let $b(\{0\} \times (0,1)) = g_1(\{1\} \times (0,1)) = g_2(\{2\} \times (0,1)) = 1$. Let $m = 1$. Even as $n \to \infty$, we will need to accept points from both the pieces $\{1\} \times (0,1)$ and $\{2\} \times (0,1)$ in order to accept both good distributions.

### 5.5.2 Multiple Samples

When $m > 1$ and there are multiple bad distributions, it turns out that sometimes the *agent* faces an NP-hard problem. This is because an individual sample may rule out several bad distributions, and to convince the principal to accept, the agent may have to judiciously choose his $m$ reported samples to *cover* all the bad distributions, in terms of ruling them out. The following theorem makes this precise.

**Theorem 5.5.** *With one good distribution and $k$ bad distributions, it is NP-hard for the agent to determine, given dataset $D$, whether it is possible to report $R \subseteq D$, such that the optimal policy accepts $R$.*

*Proof.* We reduce from the decision version of Set Cover. Given a set cover instance with elements $U$, $n'$ sets $\{S_j\}_{j\in[n']}$, and a target number $m'$, we know it is NP-hard to decide whether all elements can be covered with $m'$ sets.

We construct a strategic sample selection instance as follows. We partition the sample space into $n'$ pieces. Each piece $P_j$ corresponds to a set $S_j$ in the set cover instance. The good distribution $g$ is the uniform distribution. For each element $i \in U$, we create a bad distribution $b_i$. We set the probability density of $b_i$ to 0 on $P_j$ if $S_j \ni i$, and set it equally on all other pieces. (W.l.o.g., we can assume there is no element that is contained in all sets.)

Consider an agent with $n = n'$ samples, one for each piece. Let $m = m'$ be the number of samples he can report. Suppose that the principal will accept if and only if she is sure the underlying distribution is $g$ (say she has extremely negative utility for accepting a bad type).

Suppose a set cover of size $m$ exists. Then, the agent can report the corresponding samples. For each $b_i$, there is a sample in the report that has probability $0$ under $b_i$. Hence the principal can rule out every bad distribution.

Conversely, suppose the agent has a report that will get accepted. For each $b_i$, there must be a sample in the report that has probability density $0$ under $b_i$; this sample corresponds to some $S_j \ni i$. Hence, the agent's report produces a set cover of size $m$. □

## 5.6 Multiple Good/Bad Distributions

We now allow multiple good and multiple bad distributions. The hardness result from Theorem 5.5 still applies here when $m > 1$, so we focus on $m = 1$. Even so, Example 5.7 shows that we will get nonlocality in the optimal policy, so we do not prove a structural result. This leaves the question of whether we can efficiently compute policies with target accepting probabilities when $m = 1$.

**Theorem 5.6.** *Fix $m = 1$, $n \geq 1$, and a partition of the sample space $X$ into $t$ pieces. Assume we are given distributions $g_1, \ldots, g_k$, and $b_1, \ldots, b_\ell$ such that every distribution is constant on every piece. Then, given a vector of target accepting probabilities $(p_{g_1}, \ldots, p_{g_k}, p_{g_1}, \ldots, p_{b_\ell})$, we can decide in $\mathrm{mathrmpoly}(k, \ell, t, n)$ time whether there is a policy that can achieve these requirements.*

We defer the proof of Theorem 5.6 to the appendix.

## 5.7 Conclusion

We have introduced the problem of designing an optimal classification policy when the samples are selected by a strategic agent who favors a specific outcome.

We proved several basic structural results. If the principal aims to maximize expected utility, where she associates utilities with individual outcomes, then there is no benefit to randomization (Corollary 5.1). When distinguishing a single good from a single bad distribution, if only a single sample is reported, then the optimal policy is to accept samples whose good/bad likelihood ratio exceeds some threshold (Theorem 5.1). Moreover, in the limit as $n \to \infty$, our success probability is solely determined by the highest likelihood ratio in the sample space (Corollary 5.2). While a result similar to Theorem 5.1 holds when $m = n$ (Theorem 5.2), unfortunately nothing like it holds for the case $1 < m < n$ (Proposition 5.3). Still, we can design a policy that has good behavior in the limit for this case (Theorem 5.3). Moving on to the case of multiple bad distributions, we show that for $m = 1$, in the limit our optimal policy focuses on a single piece (Theorem 5.4)—but this is not true with multiple good distributions (Example 5.7).

We also proved basic computational results. In the discrete, deterministic case, determining whether a combination of a given false positive and a given false negative rate can be obtained is NP-hard even with $m = n = 1$ (Proposition 5.2). However, if we restrict ourselves to piecewise-constant distributions, then we can obtain an efficient algorithm even with multiple good and bad

distributions (but still $m = 1$; Theorem 5.6). When $1 < m < n$ and there are multiple bad distributions, the agent's problem of best-responding to the optimal policy becomes NP-hard (Theorem 5.5).

There are several open questions. Perhaps the most significant open questions are in the setting where there is a single good and a single bad distribution, and $1 < m < n$. We have shown that for optimal policies in this case, it is not always true that the agent should just report the "best" samples according to a single criterion. Still, do optimal policies in this case have some natural structure? Can they be computed efficiently?

## 5.8 Missing Proofs from Section 5.3

*Proof of Proposition 5.1.* For a randomized policy $\Pi_r$, consider the following distribution $\Pi_d$ over deterministic policies: Let $\Pi_d = \Pi_d(q)$ where $q \sim U[0,1]$ is a uniformly random number from $[0,1]$, and $\Pi_d(q)$ is a deterministic policy that accepts the following reports

$$\mathcal{S}_d(q) = \{R \mid \Pi_r(R) \geq q\}.$$

For any report $R$, $\Pr_{\Pi_d}(\Pi_d(R) = 1) = \Pr_q[R \in \mathcal{S}_d(q)] = \Pi_r(R)$. In other words, when the principal runs $\Pi_d$, any report $R$ is accepted with probability $\Pi_r(R)$. □

*Proof of Corollary 5.1.* Take any optimal policy $\Pi^\star$. If $\Pi^\star$ is deterministic then we are done, otherwise we decompose $\Pi^\star$ into a distribution $\Pi_d$ over deterministic policies using Proposition 5.1. The distribution $\Pi_d$ is highly structured: every deterministic policy in the support of $\Pi_d$ orders the space of reports by $\Pi_r(R)$, and accepts some prefix of this order. The agent can best respond to all these deterministic policies simultaneously, by submitting the report $R$ with the highest $\Pi_r(R)$.

This reporting strategy is optimal against all deterministic policies in the support of $\Pi_d$. As a result, if the principal switches from $\Pi^\star$ to $\Pi_d$, all reports are accepted with the same probability, and therefore, the principal's expected utility remains the same. Because the utility of $\Pi_d$ is a weighted average of the utilities of deterministic policies, one of these deterministic policies must be optimal. □

*Proof of Proposition 5.2.* Consider an instance of the Knapsack problem: we are given $s$ items with weights $w_1, \ldots, w_s$ and values $v_1, \ldots, v_s$, a maximum weight $W$, and a minimum value $V$. We are asked whether there is a subset $T$ of $\{1, \ldots, s\}$ such that $\sum_{i \in T} w_i \leq W$ and $\sum_{i \in T} v_i \geq V$. By normalization, w.l.o.g, we may assume $\sum_{i=1}^{s} w_i = \sum_{i=1}^{s} v_i = 1$.

We reduce this to an instance of our problem with $n = m = 1$ as follows. Let the sample space be $\{1, \ldots, s\}$. Let $g(i) = v_i$ and $b(i) = w_i$. Let $p_g = V$ and $p_b = W$. A deterministic policy will accept a subset $T \subseteq \{1, \ldots, s\}$. Then, the probability that we accept $g$ is $\sum_{i \in T} v_i$ and the probability that we accept $b$ is $\sum_{i \in T} w_i$. Hence, the two instances are equivalent. □

## 5.9 Proof of Theorem 5.2

*Proof of Theorem 5.2.* Suppose there are two reports $R_1 \in \mathcal{S}$ and $R_2 \notin \mathcal{S}$, both are not on the boundary of $\mathcal{S}$, such that

$$\prod_{x \in R_1} g(x)/b(x) < \prod_{x \in R_2} g(x)/b(x).$$

For notational convenience, in this proof we view a report as a sequence of samples (rather than a multiset). We consider only policies that are permutation-invariant, i.e., if a policy accepts a vector, it also accepts any permutation of that vector. Because a permutation-invariant vector-based policy is equivalent to a multiset-based policy and $m = n$, we can assume the agent always submits her entire data in the order she receives them. It follows that the (conditional) density of a report $R$ is precisely $\prod_{x \in R} \theta(x)$, where $\theta \in \{g, b\}$.

Suppose all entries of $R_1$ and $R_2$ are distinct.[9] Pick neighborhoods $\mathcal{N}_1$ of $R_1$ and $\mathcal{N}_2$ of $R_2$, such that

$$\Pr_{R \sim g^m}[R \in \mathcal{N}_1] = \Pr_{R \sim g^m}[R \in \mathcal{N}_2] > 0,$$
$$\Pr_{R \sim b^m}[R \in \mathcal{N}_1] > \Pr_{R \sim b^m}[R \in \mathcal{N}_2],$$

and the permutations of all vectors in $\mathcal{N}_1$ (resp. $\mathcal{N}_2$) do not overlap. Such neighborhoods exist because of continuity of $g$ and $b$ and our assumption on $R_1$ and $R_2$.

Now consider the permutation closures $\mathcal{P}_1$ of $\mathcal{N}_1$ (resp. $\mathcal{P}_2$ for $\mathcal{N}_2$)[10]. Because the permutations of $\mathcal{N}$ do not overlap, we have $\Pr[R \in \mathcal{P}_1] = m! \cdot \Pr[R \in \mathcal{N}_1]$. Therefore,

$$\Pr_{R \sim g^m}[R \in \mathcal{P}_1] = \Pr_{R \sim g^m}[R \in \mathcal{P}_2] > 0,$$
$$\Pr_{R \sim b^m}[R \in \mathcal{P}_1] > \Pr_{R \sim b^m}[R \in \mathcal{P}_2].$$

Note that the policy $(\mathcal{S} \setminus \mathcal{P}_1) \cup \mathcal{P}_2$ is permutation-invariant, since $\mathcal{S}$, $\mathcal{P}_1$ and $\mathcal{P}_2$ are all permutation-invariant. A similar argument as the one used in the proof of Theorem 5.1 shows that the policy $(\mathcal{S} \setminus \mathcal{P}_1) \cup \mathcal{P}_2$ is better than $\mathcal{S}$. □

## 5.10 Proof of Theorem 5.3

*Proof of Theorem 5.3.* Fix any $0 < \varepsilon < 1$. Because $g(x)/b(x)$ is a continuous function with supreme $r$, there exists some $S' \subseteq X$ such that $g(x)/b(x) \geq (1 - \varepsilon)r$ for all $x \in S'$. Let $\delta' = g(S')$. For sufficiently large $n$, we can pick a subset $S \subseteq S'$ such that $\delta = g(S) = \frac{m\sqrt{r}}{n}$. Because $S \subseteq S'$, we know that $b(S) \leq \delta/(r(1-\varepsilon))$. Similar to the proof of Corollary 5.2, we will eventually let $\varepsilon \to 0$, so for convenience we continue the proof assuming $b(S) = \delta/r = \frac{m}{\sqrt{r}n}$.

[9] By the continuity of $g(x)/b(x)$, we can always pick $R_1'$ (resp. $R_2'$) in a small neighborhood of $R_1$ (resp. $R_2$), such that all conditions specified in the theorem are satisfied and all entries of $R_1'$ (resp. $R_2'$) are distinct.

[10] The permutation closure of $\mathcal{N}$ is the set of all permutations of all vectors in $\mathcal{N}$.

Consider the policy that accepts the report iff all $m$ samples are in $S$. We chose the values of $g(S)$ and $b(S)$ so that in expectation, if the agent takes $n$ samples from $g$, $m\sqrt{r}$ of them will be in $S$; and for $n$ samples from $b$, $(m/\sqrt{r})$ of them will be in $S$. The rest of the proof uses the (multiplicative) Chernoff bound to show that the actual number of samples in $S$ concentrations around its expectation.

We first consider the probability of rejecting $g$. Let $Y_1, \ldots, Y_n$ be binary random variables such that $Y_i = 1$ iff the $i$-th sample drawn from $g$ is in $S$. Observe that $Y_i$ are i.i.d. Bernoulli random variable where $\Pr[Y_i = 1] = \delta$ and $\mathbb{E}\left[\sum_{i=1}^{n} Y_i\right] = mr^{1/2}$. By the Chernoff bound,

$$\begin{aligned}
\Pr[\text{reject } g] &= \Pr\left[\sum_{i=1}^{n} Y_i < m\right] \\
&= \Pr\left[\sum_{i=1}^{n} Y_i < r^{-1/2} \cdot \mathbb{E}\left[\sum_{i} Y_i\right]\right] \\
&\leq \exp\left(-\frac{1}{2}\left(1 - r^{-1/2}\right)^2 r^{1/2} m\right).
\end{aligned}$$

Similarly, for $n$ samples drawn from $b$, let $Z_1, \ldots, Z_n$ be Bernoulli random variable where $Z_i = 1$ iff the $i$-th sample is in $S$. Note that $\Pr[Z_i = 1] = \delta/r$ and $\mathbb{E}\left[\sum_{i=1}^{n} Z_i\right] = mr^{-1/2}$. Again by Chernoff bound,

$$\begin{aligned}
\Pr[\text{accept } b] &= \Pr\left[\sum_{i=1}^{n} Z_i \geq m\right] \\
&= \Pr\left[\sum_{i=1}^{n} Z_i \geq r^{1/2} \cdot \mathbb{E}\left[\sum_{i} Z_i\right]\right] \\
&\leq \exp\left(-\frac{1}{r^{1/2}+1}\left(r^{1/2} - 1\right)^2 r^{-1/2} m\right).
\end{aligned}$$

The theorem follows from taking the maximum of the two upper bounds on the error probability. □

## 5.11 Proof of Theorem 5.6

*Proof of Theorem 5.6.* We use $P_1, \ldots, P_t$ to denote the $t$ pieces and assume w.l.o.g. that each piece has measure $|P_i| = 1$. Let $\alpha_j^{\theta}$ be the density of distribution $\theta$ on $P_j$. For example, the "cumulative" probability of $g_i$ on a subset $S \subseteq P_j$ is $\alpha_j^{g_i}|S|$.

We will write a mathematical program to decide whether the target accepting probabilities are achievable. The variables $x_j$ denote the fraction of piece $j$ that a policy $\Pi$ will accept. We can

write out the accepting probabilities of $\Pi$ explicitly, and put constraints on them. [11]

$$\begin{aligned}
&1 - \left(1 - \textstyle\sum_j \alpha_j^{g_i} x_j\right)^n \geq p_{g_i}, && \forall i \in [k], \\
&1 - \left(1 - \textstyle\sum_j \alpha_j^{b_i} x_j\right)^n \leq p_{b_i}, && \forall i \in [\ell], \\
&0 \leq x_j \leq 1, && \forall j \in [t].
\end{aligned}$$

Observe the above is equivalent to the following linear program (LP):

$$\begin{aligned}
&1 - \textstyle\sum_j \alpha_j^{g_i} x_j \leq (1 - p_{g_i})^{1/n}, && \forall i \in [k], \\
&1 - \textstyle\sum_j \alpha_j^{b_i} x_j \leq (1 - p_{b_i})^{1/n}, && \forall i \in [\ell], \\
&0 \leq x_j \leq 1, && j \in [t].
\end{aligned}$$

The theorem follows immediately from the fact that we can write down this LP and check its feasibility in $\mathrm{mathrmpoly}(k, \ell, t, n)$ time. $\square$

[11] we use $[n]$ to denote the set of integers $\{1, \ldots, n\}$.

# Chapter 6

# Distinguishing Distributions When Samples Are Strategically Transformed

## 6.1 Introduction

In Chapter 5, we have discussed classification based on strategically selected samples. In this chapter, we investigate an equally natural and important setting, where agents can *transform* private samples into signals, which are to be reported to the principal.

Why should we consider classification based on transformed samples? First, anyone can have a bad day, or a lucky one. Thus, in general, to determine with reasonable confidence who are the highly capable agents—whether they be people, companies, or anything else—we need to observe their output over an extended period of time. Moreover, capability is generally not one-dimensional, and who should be considered highly capable depends on what it is that we are looking for. Finally, the policy that we set to evaluate agents' output will in general affect how they strategically try to shape that output. Thus, we must choose our policy to enable the agents that are highly capable (according to our definition) to distinguish themselves from others.

**Example.** Suppose that there are researchers of different *types*. Specifically, suppose we have the following set of types:

$$\Theta = \{\text{TML-H}, \text{TML-L}, \text{AML-H}, \text{AML-L}\}$$

where "TML" stands for "theoretical machine learning," "AML" for "applied machine learning," and "L" and "H" for "low quality" and "high quality," respectively. Each researcher generates high-quality *ideas* (which we will in this chapter refer to as *samples*) according to some probabilistic process. Suppose here the sample space is

$$S = \{\text{T}, \text{A}, \text{B}\}$$

where "T" stands for a purely theoretical idea without immediate applied significance, "A" for an applied idea without immediate theoretical significance, and "B" for an idea that has both theoretical and applied significance. Finally, suppose there are only 3 conferences: COLT, KDD, and

NeurIPS (we will in this chapter refer to papers published in these conferences as "signals").

$$\Sigma = \{\text{COLT}, \text{KDD}, \text{NeurIPS}\}$$

A T or a B idea (sample) can be turned into a COLT paper (signal);[1] an A or a B idea can be turned into a KDD paper; and a T, A, or B idea can be turned into a NeurIPS paper.[2] Each idea, of course, can be published in only one conference.

Suppose a university would like to hire an AML-H researcher (but none of the other types). The faculty recruiting committee, unfortunately, is excessively lazy and only looks at the publication counts in the various venues. While the candidate researchers of course are committed to improving this terrible process once they get the job, for now their only concern is getting the job. In particular, everyone will attempt to pretend to be an AML-H researcher by sending their papers to the appropriate venues. But what exactly does this mean?

Suppose an AML-H researcher generates ideas at the following rates: $0.5$ B, $0.4$ A, $0.1$ T. Moreover suppose that a TML-H researcher generates ideas at the following rates: $0.5$ B, $0.1$ A, $0.4$ T. If the AML-H researcher sends all her papers to NeurIPS, then, even in the long run, she cannot distinguish herself from the TML-H researcher, who could do the same. On the other hand, if she sends strictly more than $0.6$ of her ideas to KDD, then in the long run she will be able to distinguish herself from the TML-H researcher, because $0.4$ of the latter's ideas cannot go to KDD.

Now consider the AML-L researcher. First, an easy case: suppose he generates ideas at the following rates: $0.4$ B, $0.3$ A, $0$ T. (These numbers do not sum to $1$, but this is not necessary, since they are rates. Equivalently, we can suppose him to have "the empty idea" $\emptyset$ with the remaining probability $0.3$, which can be sent only to "the empty conference" where anything can be sent. This "empty signal" can also be used to model that the researchers sometimes only have ideas that they do not consider worth publishing, i.e., that they strategically select only a subset of their samples to pursue.) Clearly the AML-H researcher will in the long run distinguish herself from the AML-L researcher simply by the overall number of papers published (as long as the AML-H researcher does not unnecessarily send papers to the empty conference!). Alternatively, suppose the AML-L researcher generates ideas at the following rates: $0.4$ B, $0.5$ A, $0.1$ T (so that the only weakness of the AML-L researcher relative to the AML-H researcher is that fewer of his ideas have both theoretical and applied significance). In this case, the AML-H researcher can, in the long run, distinguish herself from the AML-L researcher by sending strictly more than $0.5$ of her ideas to COLT. Of course, this conflicts with what she needs to do distinguish herself from the TML-H researcher. Still, she can distinguish herself from both the TML-H and the AML-L researcher in the long run by, in odd-numbered years, sending strictly more than $0.6$ of her ideas to KDD, and, in even-numbered years, sending strictly more than $0.5$ of her ideas to COLT.

In the long run we are all dead. —*John Maynard Keynes*

[1] Of course, having the basic idea is generally only a small part of the work that needs to be done for a conference paper; but for our purposes here, we may imagine that the idea incorporates all the work that needs to be done.

[2] We use the names of actual conferences strictly for amusement value, and while we think our example roughly aligns with the focus of these conferences, we do not mean to imply anything about their selectivity (all these ideas are high-quality) or open-mindedness. We also do not mean to imply anything about other conferences—e.g., ICML could just as well have been used instead of NeurIPS—or (in what follows) about different types of researchers or the priorities and effort levels of actual hiring committees.

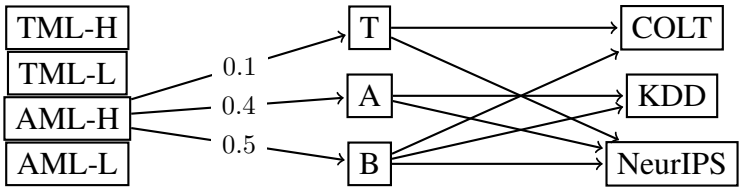


Figure 6.1: Illustration of the example.

In reality, the candidates will have only finite time to prove themselves. Still, the lazy committee may hope to distinguish them with high probability. How many years suffice for this (and, therefore, should be the length of a typical Ph.D. program, potentially extended with a postdoctoral appointment)?

While this is example is a bit tongue-in-cheek, it is not hard to see that this basic phenomenon frequently occurs in society. People select from their opportunities and craft them to fit what they think will appeal to future employers. A start-up company may select from its opportunities and craft them to fit what they think will impress future backers. In this chapter, we introduce a general model that captures all these and other cases. Within this model, we characterize conditions under which agents of certain types can distinguish themselves from others, as well as how many samples are needed for this.

## 6.2 Preliminaries

For a set $S$, we use $\Delta(S)$ to denote the set of probability distributions over $S$. Given a distribution $x \in \Delta(S)$, we use $x(i)$ to denote the probability mass on the element $i \in S$, and $x(A)$ to denote the total probability mass on the set $A \subseteq S$. We are generally interested in distinguishing one or more *good* distributions from one or more *bad* distributions (where good and bad are determined by what we are looking for). We use $g$ to denote the good distribution, and $b$ to denote the bad distribution. (We use $(g_i)_i$ and $(b_i)_i$ when there are multiple good/bad distributions.) The agent, depending on his type being either good or bad, draws $n$ samples i.i.d. from either $g$ or $b$. How samples can be turned into signals is represented by a bipartite graph $G = (S \cup \Sigma, E)$ between the (discrete) sample space $S$ and the (discrete) signal space $\Sigma$. An agent must convert each sample into a signal and then submit all $n$ signals to the principal. $E$ specifies which signals are valid for each sample: a sample $s \in S$ can be converted into a signal $\sigma \in \Sigma$ iff $(s, \sigma) \in E$.

Note that our model generalizes each of the following models:

1. The agent can choose to omit samples. We can add an "empty signal" to $\Sigma$, where converting a sample $s$ to the empty signal corresponds to not reporting $s$.
2. The agent may or may not receive a sample in each round. E.g., in the example where samples correspond to ideas and signals correspond to papers, in some rounds the agent may not have any (worthwhile) idea. We can add an "empty sample" in $S$ which can only be converted to the empty signal.
3. The signal space is the same as the sample space: $S = \Sigma$. In this case it is more natural to

replace the bipartite graph by one that has only one copy of each sample/signal, is no longer bipartite, and that represents the possibility of changing sample/signal $u$ to sample/signal $v$ by a directed edge $(u, v)$.

We will be interested in the probability of accepting good or bad types after $T$ rounds (i.e., after the agent draws $T$ samples). We call the $T$ signals submitted a *report* $\mathcal{R} \in \Sigma^T$. The principal gets to choose an acceptance function (or policy, which could be randomized) $f : \mathcal{R} \to \{0, 1\}$ that maps the report into a binary decision. Her goal is to accept the good agent and reject the bad agent. The agent wants to be accepted regardless of his type. The principal can thus make two types of mistakes: false-positive (or type 1 error) when she accepts a bad agent, and false-negative (type 2 error) when she rejects a good agent. The principal wants to minimize the maximum probability of making either type of mistakes.

We recall the following definition of the total variation distance:

**Definition 6.1** (Total Variation Distance)**.** The total variation distance between two distributions $x, y \in \Delta(\Sigma)$ over support $\Sigma$ is defined to be

$$d_{\mathrm{TV}}(x, y) = \frac{1}{2}\|x - y\|_1 = \frac{1}{2}\sum_{\sigma\in\Sigma} |x(\sigma) - y(\sigma)| = \max_{A\subseteq\Sigma}(x(A) - y(A)).$$

In our setting, the total variation distance provides a good way to measure the closeness between two *signal* distributions, which are observable by the principal. We will generalize this definition to our strategic setting, to measure how close two distributions over the *sample* space are to each other.

## 6.3 Basic Structural Results

In this section, we define a notion that we term "directed total variation distance" $d_{\mathrm{DTV}}$. For two distributions $x$ and $y$ over samples, $d_{\mathrm{DTV}}(x, y)$ measures how well $x$ can distinguish itself from $y$ in our strategic setting. As we will see in the later sections, $d_{\mathrm{DTV}}$ is a central notion in this chapter, and often dictates the number of samples we need to distinguish the two distributions under strategic reporting.

We first give the formal definitions of *reporting strategies* and the directed total variation distance $d_{\mathrm{DTV}}(x, y)$. Then we define another notion $\mathrm{MaxSep}(x, y)$ that measures how well $x$ can distinguish itself from $y$ from the principal's perspective, using separating sets instead of reporting strategies. Given these definitions, we present one of our key structural results (Proposition 6.1), which shows that the two notions are equivalent.

Before investigating distinguishing distributions under strategic reporting, we first generalize the classical measure of how close two distributions are, $d_{\mathrm{TV}}$, to our strategic setting. We first give a formal definition the *reporting strategy* used by the agents.

**Definition 6.2** ((Single-Round) Reporting Strategy)**.** Given $x \in \Delta(S)$, $\alpha \in \Delta(\Sigma)$, we say $x$ *can report* $\alpha$ ($x \to \alpha$), if there exist a *reporting strategy* $R = \{r_{s,\sigma}\}_{(s,\sigma)\in E}$ satisfying:

- $r_{s,\sigma} \geq 0$ for all $(s, \sigma) \in E$.
- For each $s \in S$, $\sum_{\sigma:(s,\sigma)\in E} r_{s,\sigma} = 1$.

- For each $\sigma \in \Sigma$, $\sum_{s:(s,\sigma)\in E} x(s) \cdot r_{s,\sigma} = \alpha(\sigma)$.

We say $x$ reports $\alpha$ by strategy $R$ ($x \to_R \alpha$).

In other words, when each sample $s \in S$ is drawn from the distribution $x$ and given this sample the agent is reporting $\sigma \in \Sigma$ with probability $r_{s,\sigma}$, the resulting distribution over the signal space is exactly $\alpha$. For a fixed sample or a random variable $s$, we use $R(s) \in \Delta(\Sigma)$ to denote the random variable whose distribution over the signal space is induced by $\{R_{s,\sigma}\}_{\sigma\in\Sigma}$.

Given the definition of reporting strategies, we are ready to generalize $d_{\mathrm{TV}}$ to our setting. Intuitively, $x$ chooses a report first, and then $y$ chooses a report in response; they play a zero-sum game where $x$ wants the reports to be as far away from each other as possible. $d_{\mathrm{DTV}}(x, y)$ is the value of this two-player game when $x$ must choose a report (i.e., a pure strategy) first, which measures how far $x$ can stay away from $y$.

**Definition 6.3** (Directed Total Variation Distance)**.** Given $(S, \Sigma, E)$, the directed total variation distance between two distributions $x, y \in \Delta(S)$ over the sample space $S$ is defined to be

$$d_{\mathrm{DTV}}(x, y) = \max_{\alpha:x\to\alpha} \min_{\beta:y\to\beta} d_{\mathrm{TV}}(\alpha, \beta).$$

Directed total variance distance nicely characterizes the distance between two distributions from the agent's perspective, but it is not immediately clear how that might help the principal. In particular, are two distributions easily separable by setting an appropriate policy if they have large directed total variation distance? To study this, we introduce several concepts to model the problem from the principal's perspective.

**Definition 6.4** (Preimage of Signals)**.** For any set of signals $A \subseteq \Sigma$, the preimage $\mathrm{pre}(A)$ of $A$ is defined to be the set of samples which can be mapped to a signal in $A$. That is

$$\mathrm{pre}(A) = \{s \in S \mid \exists \sigma \in A, \text{ s.t. } (s, \sigma) \in E\}.$$

The principal could label a set $A$ of signals as "good" signals and simply measure how many good signals the agent is able to send. Ideally, this $A$ is chosen so that a good agent can send (significantly) more signals in $A$ than a bad agent. This inspires the following definitions.

**Definition 6.5** (Separation)**.** For any $A \subseteq \Sigma$, if $x(\mathrm{pre}(A)) - y(\mathrm{pre}(A)) = \varepsilon > 0$, then we say $A$ separates $x$ from $y$ by a margin of $\varepsilon$.

**Definition 6.6** (Max Separation)**.** The max separation of $x \in \Delta(S)$ from $y \in \Delta(S)$ over the sample space $S$ is defined to be $\mathrm{MaxSep}(x, y) = \max_{A\subseteq\Sigma}(x(\mathrm{pre}(A)) - y(\mathrm{pre}(A)))$.

We now draw the connection between the agent's and the principal's perspectives. The following proposition can be viewed as a generalization of the classic Hall's Marriage Theorem. Proposition 6.1 states that $g$ can distinguish itself from $b$ under strategic reporting iff there exists a subset $A^*$ of signals so that $g$ can generate more signals in $A^*$ than $b$. Equivalently, the best reporting strategy for $g$ is to focus on a subset $A^*$ of the signal space, and try to convert samples into signals in $A^*$ whenever possible.

**Proposition 6.1.** *For any $x, y \in \Delta(S)$, $d_{\mathrm{DTV}}(x, y) = \mathrm{MaxSep}(x, y)$.*

The proof of the proposition, as well as all other proofs, is deferred to the appendix. This equivalence between $d_{\mathrm{DTV}}$ and MaxSep not only is a nice structural result; Proposition 6.1 plays a substantial part in our main algorithmic results.

It is worth noting that $d_{\mathrm{DTV}}(x, y)$ in general is not equal to $d_{\mathrm{DTV}}(y, x)$. However, the triangle inequality still holds for $d_{\mathrm{DTV}}$, which also enables some of our main results.

**Proposition 6.2.** *For any $x, y, z \in \Delta(S)$, $d_{\mathrm{DTV}}(x, y) + d_{\mathrm{DTV}}(y, z) \geq d_{\mathrm{DTV}}(x, z)$.*

## 6.4 Structural and Computational Results in the General Case

In this section, we define adaptive and non-adaptive reporting strategies (Definition 6.7), and the accepting probabilities of the optimal reporting strategies after $T$ rounds (Definition 6.8). At a high level, we give a tight characterization result on when there exists a policy that can distinguish $g$ from $b$ under strategic reporting, and provide an asymptotically tight bound on the sample complexity of the optimal policy. Moreover, we show that while our structural result is clean and tight, it is computationally hard to check if the condition holds. That is, in the general case, it is NP-hard to determine whether there is a policy that can distinguish $g$ from $b$.

More specifically, we first show that there exists a policy that can distinguish $g$ from $b$ in the limit (when $T \to \infty$) iff $d_{\mathrm{DTV}}(g, b) > 0$ (Theorem 6.1). Next, we give an asymptotically tight sample complexity bound of $T = \Theta(1/\varepsilon^2)$ when $d_{\mathrm{DTV}}(g, b) = \varepsilon$ and we want to distinguish $g$ from $b$ with high constant probability (Theorem 6.3). We then extend the existence result to more general settings when there are multiple good and bad distributions (Theorem 6.4). Finally, we show that it is NP-hard to decide if we are in the case where $d_{\mathrm{DTV}}(g, b) = 0$ or $d_{\mathrm{DTV}}(g, b) > \frac{1}{\mathrm{poly}(m,n)}$ (Theorem 6.2).

We start with the definition of adaptive reporting strategies.

**Definition 6.7** (Adaptive Reporting Strategy)**.** An adaptive reporting strategy $\mathcal{R} = (R^1, \ldots, R^T)$ is a sequence of (different) reporting strategies. The signal $\sigma^i$ at time $i$ is obtained by applying $R^i$ to the sample $s^i$ at time $i$. $R^i = R^i(\sigma^1, \ldots, \sigma^{i-1})$ may depend on all past signals. A reporting strategy is non-adaptive if $R^i = R^1$ for any $i$ and $(\sigma^1, \ldots, \sigma^{i-1})$, and adaptive otherwise. For an adaptive policy $\mathcal{R} = (R^1, \ldots, R^T)$, we interchangeably write $\sigma^i = R^i(s^i \mid \sigma^1, \ldots, \sigma^{i-1})$ to indicate the dependence of $R^i$ on $\sigma^1, \ldots, \sigma^{i-1}$.

When we analyze the quality of a fixed $T$-round policy $f$, we are interested in the probability that $f$ accepts $g$ or $b$ after $T$ rounds, when the agent (of either type) best-responds to $f$.

**Definition 6.8** (Acceptance Probabilities of the Best Reporting Strategies)**.** Given $x \in \Delta(S)$, $T \in \mathbb{N}$, and the principal's policy $f$, let the acceptance rate under adaptive / non-adaptive reporting respectively be

$$p_{\mathrm{ada}}(f, x, T) = \max_{\mathcal{R}=(R^1,\ldots,R^T)} \mathbb{E}[f((R^i(s^i))_{i\in[T]})],$$
$$p_{\mathrm{non}}(f, x, T) = \max_{\mathcal{R}=(R,\ldots,R)} \mathbb{E}[f((R^i(s^i))_{i\in[T]})]$$

where the expectations are taken over $T$ i.i.d. samples $(s^i)_i$ drawn from $x$. Observe that $p_{\mathrm{ada}}(f, x, T) \geq p_{\mathrm{non}}(f, x, T)$ for any $f$, $x$ and $T$.

Intuitively, if $d_{\mathrm{DTV}}(g, b) = 0$, then the bad distribution can mimic the good distribution perfectly in the signal space, no matter what reporting strategy $g$ uses. Therefore, it is impossible to distinguish $g$ from $b$. The next theorem formalizes this intuition. In particular, even if $g$ reports

adaptively, $b$ can still mimic $g$'s conditional reporting strategy in every situation (i.e., for every combination of previously reported signals).

**Theorem 6.1** (Separability in the Limit)**.** *Given good and bad distributions $g$ and $b$:*

*(i) If $d_{\mathrm{DTV}}(g,b) > 0$, then there exists a policy $f$ such that*

$$\lim_{T\to\infty}\left(p_{\mathrm{non}}(f,g,T) - p_{\mathrm{ada}}(f,b,T)\right) = 1.$$

*That is, $f$ accepts $g$ and rejects $b$ with probability $1$ in the limit.*

*(ii) If $d_{\mathrm{DTV}}(g,b) = 0$, then for any policy $f$ and any $T$,*

$$p_{\mathrm{ada}}(f,g,T) \le p_{\mathrm{ada}}(f,b,T), p_{\mathrm{non}}(f,g,T) \le p_{\mathrm{non}}(f,b,T).$$

*That is, no policy can separate $g$ from $b$, regardless of whether the setting is adaptive.*

The next theorem states that while our characterization result (Theorem 6.1) is clean and tight (we can distinguish iff $d_{\mathrm{DTV}}(g,b) > 0$), it is in fact computationally hard to check if this condition holds. Intuitively, Theorem 6.2 constructs an instance where the good distribution needs to focus on as few signals as possible. The parameters are chosen carefully so that it is crucial that $g$ finds a subset of signals $A \subseteq \Sigma$ with minimum cardinality that covers the support of $g$.

**Theorem 6.2** (hardness of checking separability)**.** *Given $x, y \in \Delta(S)$, it is* NP*-hard to distinguish between the following two cases: (1) $d_{\mathrm{DTV}}(x,y) = 0$ and (2) $d_{\mathrm{DTV}}(x,y) \ge \frac{1}{\mathrm{poly}(m,n)}$, or equivalently, to determine the existence of a set $A \subseteq \Sigma$ such that $x(\mathrm{pre}(A)) - y(\mathrm{pre}(A)) \ge \frac{1}{\mathrm{poly}(m,n)}$.*

Note that the hardness of checking the existence of separating sets implies the hardness of finding any separating set given that $d_{\mathrm{DTV}}(x,y) > 0$. This is because given an algorithm for the latter problem, one could run that algorithm without knowing whether $d_{\mathrm{DTV}}(x,y) > 0$ and see if it succeeds. Either the algorithm returns a separating set, or we know it must be the case that $d_{\mathrm{DTV}}(x,y) = 0$ and no separating set exists.

Next, we focus on the case when there are finitely many samples. Theorem 6.3 is more refined than Theorem 6.1, in that it gives a tight sample complexity bound instead of only talking about distinguishing $g$ and $b$ in the limit.

**Theorem 6.3** (Sample Complexity with Two Distributions)**.** *For any $g$ and $b$ such that $d_{\mathrm{DTV}}(g,b) \ge \varepsilon$:*

- *There is a policy $f$ such that for any $\delta > 0$ and $T \ge 2\ln(1/\delta)/\varepsilon^2$, $p_{\mathrm{non}}(f,g,T) \ge 1-\delta$ and $p_{\mathrm{ada}}(f,b,T) \le \delta$.*
- *When $d_{\mathrm{DTV}}(g,b) = \varepsilon$ and $T = o(1/\varepsilon^2)$, for any $f$, $p_{\mathrm{non}}(f,g,T) - p_{\mathrm{non}}(f,b,T) < \frac{1}{3}$.*

Theorem 6.3 can be generalized to the case where there are multiple good and bad distributions. First, suppose there is one good distribution and multiple bad distributions. As long as $d_{\mathrm{DTV}}(g,b_j) \ge \varepsilon$ for every bad distribution $b_j$, we can use the testing algorithm in Theorem 6.3 to distinguish them in $T = O(1/\varepsilon^2)$ rounds (with high constant probability). We potentially need to do so separately for every bad distribution, paying an extra factor of $\Omega(\ell)$ in the sample complexity if there are $\ell$ bad distributions. If there are $k$ good distributions, then we can run the $k$ testers in parallel, paying an additional factor of $\log(k)$ in the sample complexity to boost the success probability so that we can take a union bound.

**Theorem 6.4** (Multiple Good and Bad Distributions, the General Case)**.** *For any $g_1, \dots, g_k$ and $b_1, \dots, b_\ell$ such that $d_{\mathrm{DTV}}(g_i, b_j) \geq \varepsilon$ for any $i \in [k]$ and $j \in [\ell]$, there is a policy $f$ such that: For any $\delta > 0$ and $T \geq 2\ell \ln(k\ell/\delta)/\varepsilon^2$, $p_{\mathrm{ada}}(f, g_i, T) \geq 1 - \delta$ for any $i \in [k]$, and $p_{\mathrm{ada}}(f, b_j, T) \leq \delta$ for any $j \in [\ell]$.*

We note that the policy in Theorem 6.4 requires the good distribution to report in different ways, which is not possible with a non-adaptive strategy according to our definition. In particular, the good distribution must know which bad distribution it is up against in each phase, and report accordingly. As our introductory example shows, this is in fact necessary when there are multiple bad distributions.

## 6.5 When Signals Are Partially Ordered

In many real-world situations, the sample and signal spaces are structured. For example, when a band is recruiting new members, applicants may be asked to submit video recordings of themselves playing. An applicant would probably videotape herself playing for an entire event as a sample, and then crop the recording to create a signal that demonstrates only her best performance. This cropping procedure is irreversible: the complete recording may be cropped to keep a part, but from a part, it is impossible to recover the full recording. The signal space in this scenario is partially ordered by the cropping procedure—the samples/signals can be transformed in one direction (shortening), but never the other. Also, there is a "default" signal for each sample, which is simply to submit the complete recording without cropping. The default signal can be transformed into any signal that can be reported from this sample. In this section, we consider the following abstraction of such scenarios:

- $S = \Sigma$,
- $(s, s) \in E$ for any $s \in S$,
- $(s, t) \in E$ and $(t, u) \in E \implies (s, u) \in E$, and
- $E$ is acyclic except for self-cycles.

This abstraction also covers, for example, scenarios where the agent can choose to hide certain samples—any sample can be transformed into a non-sample, but not reversely. Note that given the above conditions, the sample/signal space is essentially a partially ordered set, where a sample can only be transformed according to this partial order. Let $n = |S|$ be the cardinality of the sample/signal space.

We first show some useful structural results in the partially ordered case. The following proposition demonstrates that the revelation principle holds in this case.

**Proposition 6.3** (Revelation Principle)**.** *For any policy $f$:*

- *There exists a policy $f'$ such that for any $x \in \Delta(S)$, $T \in \mathbb{N}$,*

$$p_{\mathrm{non}}(f, x, T) = p_{\mathrm{non}}(f', x, T) = \mathbb{E}[f'((s^i)_i)].$$

- *There exists a policy $f''$ such that for any $x \in \Delta(S)$, $T \in \mathbb{N}$,*

$$p_{\mathrm{ada}}(f, x, T) = p_{\mathrm{ada}}(f'', x, T) = p_{\mathrm{non}}(f'', x, T) = \mathbb{E}[f''((s^i)_i)].$$

In other, non-learning contexts in mechanism design, whether the revelation principle holds is often an aspect that determines whether the computational problems therein are tractable. We will see that this is also the case for our problem—the revelation principle enables efficient computation of the max separation, and therefore efficient policies in a quite natural way.

The next proposition simplifies the definition of $d_{\mathrm{DTV}}$ in the partially ordered case, based on the insight that, per the revelation principle, the best way for $x$ to avoid being mimicked by $y$ is to always report the unmodified samples.

**Proposition 6.4** ($d_{\mathrm{DTV}}$ Simplified)**.** *In the transitive case,* $d_{\mathrm{DTV}}(x, y) = \min_{y \to y'} d_{\mathrm{TV}}(x, y')$.

This also gives us an efficient algorithm for finding the set that supports the max separation $\mathrm{MaxSep}(x, y)$ of $x$ from $y$:

**Corollary 6.1** (Efficient Computation of Max Separation)**.** *Given any* $x, y \in \Delta(S)$*, there is a poly-time algorithm which computes a set* $A^*$ *satisfying* $x(\mathrm{pre}(A^*)) - y(\mathrm{pre}(A^*)) = \mathrm{MaxSep}(x, y)$.

We show in Theorem 6.5 that in the partially ordered case we can separate multiple good distributions from multiple bad ones with much smaller overhead. The proof of Theorem 6.5 is similar to that of Theorem 6.4. The only difference is that, because of the revelation principle, we no longer require good distributions to report adaptively.

**Theorem 6.5** (Multiple Good and Bad Distributions: The Partially Ordered Case)**.** *For any* $g_1, \ldots, g_k$ *and* $b_1, \ldots, b_\ell$ *where* $d_{\mathrm{DTV}}(g_i, b_j) \geq \varepsilon$ *for any* $i \in [k]$*,* $j \in [\ell]$*, there is a policy* $f$ *such that: For any* $\delta > 0$ *and* $T \geq 2\ln(k\ell/\delta)/\varepsilon^2$*,* $p_{\mathrm{non}}(f, g_i, T) \geq 1 - \delta$ *for any* $i \in [k]$*, and* $p_{\mathrm{ada}}(f, b_j, T) \leq \delta$ *for any* $j \in [\ell]$.

In the partially ordered case, we cannot only deal with multiple good and bad distributions much more efficiently, but also deal with any bad distribution using a single sample-efficient policy. Before stating the result, recall the following definition of the *width* of a partially ordered set.

**Definition 6.9** (Width of Partially Ordered Sets)**.** The width $\rho(G)$ of a partially ordered set represented as graph $G = (S, E)$ is defined to be $\rho(G) = \max\{|A| \mid A \subseteq S, \forall s_1, s_2 \in A, (s_1, s_2) \notin E\}$. In other words, the width is the maximum size of a set $A \subseteq S$ where any two elements in $A$ are not comparable. Such a set $A$ is called an *anti-chain*.

We now provide our generic policy, whose sample complexity, quite surprisingly, depends roughly linearly on the width of the sample space.

**Theorem 6.6** (Efficient Policy against Any Bad Distribution)**.** *For any* $g \in \Delta(S)$*, there is a policy* $f$ *such that for any* $\delta > 0$*, and* $T \geq \frac{2\rho\ln(1+n/\rho)\ln(1/\delta)}{\varepsilon^2}$*: (1)* $p_{\mathrm{non}}(f, g, T) \geq 1 - \delta$*, and (2) for any* $b$ *such that* $d_{\mathrm{DTV}}(g, b) \geq \varepsilon$*,* $p_{\mathrm{ada}}(f, b, T) \leq \delta$*. Moreover, the outcome of the policy can be computed in polynomial time.*

The above policy is able to detect any bad distribution with adaptive reporting. For bad distributions without adaptive reporting, when $\rho = \Omega(\sqrt{n}/\log n)$, the following policy achieves even better sample complexity.

**Theorem 6.7** (Efficient Policy against Non-adaptive Bad Distributions)**.** *For any* $g \in \Delta(S)$*, there is a policy* $f$ *such that for any* $\delta > 0$*, with* $T = O\left(\frac{\sqrt{n}\ln(1/\delta)}{\varepsilon^2}\right)$ *samples: (1)* $p_{\mathrm{non}}(f, g, T) \geq 1 - \delta$*, and (2) for any* $b$ *such that* $d_{\mathrm{DTV}}(g, b) \geq \varepsilon$*,* $p_{\mathrm{non}}(f, b, T) \leq \delta$*. Moreover, the outcome of the policy can be computed in polynomial time.*

## 6.6 Future research

In this chapter, we have focused on distinguishing good and bad types with near certainty. In reality, the number of available samples may not always be sufficient for this. If so, it may be worthwhile to move beyond simple acceptance and rejection decisions to a more general mechanism design setup. For example, when the signals we receive from an agent are not decisive one way or another, perhaps an intermediate outcome between rejection and acceptance allows us to improve our objective, by avoiding the damage of either accepting a bad type or rejecting a good type. One may also consider settings in which signaling is costly (or at least sending high-quality signals comes at an effort cost, in line with traditional signaling models [97]) or in which agents can in fact improve their actual types via some investment cost. Any of these directions would further enrich the specific connections between mechanism design and learning theory that we have begun to explore in this chapter (and that in turn complement other fascinating connections between these topics that have earlier been established by others [6, 14, 20, 21, 36, 57, 59, 72]).

## 6.7 Omitted Proofs From Section 6.3

We need the following fact:

**Proposition 6.5** (Saturation)**.** *If $x \to \alpha$, then for any $A \subseteq \Sigma$,*

$$x(\mathrm{pre}(A)) \geq \alpha(A).$$

*Moreover, there exists $\alpha_A$ where $x \to \alpha_A$, such that*

$$x(\mathrm{pre}(A)) = \alpha_A(A).$$

*We call the corresponding reporting strategy that achieves $x \to \alpha_A$ "saturating" for $A$.*

*Proof of Proposition 6.5.* Let $R = \{r_{s,\sigma}\}_{(s,\sigma)\in E}$ be the reporting strategy by which $x$ reports $\alpha$.

$$\begin{aligned}
x(\mathrm{pre}(A)) &= \sum_{s\in\mathrm{pre}(A))} x(s) \\
&\geq \sum_{s\in\mathrm{pre}(A)} \sum_{\sigma\in A} r_{s,\sigma} x(s) && (\textstyle\sum_{\sigma\in A} r_{s,\sigma} \leq 1) \\
&= \sum_{\sigma\in A} \sum_{s:(s,\sigma)\in E} r_{s,\sigma} x(s) \\
&= \sum_{\sigma\in A} \alpha(\sigma) && (\text{definition of } R) \\
&= \alpha(A).
\end{aligned}$$

Now we show $\alpha_A$ exists by constructing the corresponding reporting strategy. Let $R' = \{r'_{s,\sigma}\}$ be any reporting strategy satisfying: if $s \in \mathrm{pre}(A)$, $r'_{s,\sigma} = 0$ for all $\sigma \notin A$. Such an $R'$ exists

because by the definition of $\mathrm{pre}(A)$, for every $s \in \mathrm{pre}(A)$, there is at least one $\sigma \in A$ that connects to $s$.

Now for any $s \in \mathrm{pre}(A)$,

$$\sum_{\sigma \in A} r'_{s,\sigma} = 1.$$

Hence, for this reporting strategy, the single inequality in the derivation above becomes an equality, allowing us to conclude $x(\mathrm{pre}(A)) = \alpha_A(A)$. □

*Proof of Proposition 6.1.* We first show $\mathrm{MaxSep}(x, y) \leq d_{\mathrm{DTV}}(x, y)$. Let $A^* = \mathrm{argmax}_A(x(\mathrm{pre}(A)) - y(\mathrm{pre}(A)))$.

$$\begin{aligned}
d_{\mathrm{DTV}}(x, y) &= \max_{\alpha: x \to \alpha} \min_{\beta: y \to \beta} d_{\mathrm{TV}}(\alpha, \beta) \\
&\geq \max_{\alpha: x \to \alpha} \min_{\beta: y \to \beta} \sum_{\sigma \in A^*} \max\{\alpha(\sigma) - \beta(\sigma), 0\} && \text{(Definition 6.1 of } d_{TV}\text{)} \\
&\geq \max_{\alpha: x \to \alpha} \min_{\beta: y \to \beta} (\alpha(A^*) - \beta(A^*)) \\
&\geq \max_{\alpha: x \to \alpha} (\alpha(A^*) - y(\mathrm{pre}(A^*))) && \text{(Proposition 6.5)} \\
&= x(\mathrm{pre}(A^*)) - y(\mathrm{pre}(A^*)) && \text{(Proposition 6.5, existence of saturating distribution)} \\
&= \mathrm{MaxSep}(x, y).
\end{aligned}$$

Now we show $\mathrm{MaxSep}(x, y) \geq d_{\mathrm{DTV}}(x, y)$. Let $\alpha^*$ be a signal distribution reported by $x$ that achieves $d_{\mathrm{DTV}}(x, y)$. Let $\beta^*$ be a signal distribution reported by $y$ that best-response to $\alpha^*$, where we require as a tie-breaker that $\beta^*$ minimizes the number of signals $\sigma$ with $\alpha^*(\sigma) \geq \beta^*(\sigma)$.

Let $A^* = \{\sigma \mid \alpha^*(\sigma) \geq \beta^*(\sigma)\}$. We will show that $A^*$ separates $x$ from $y$ by a margin of $d_{\mathrm{DTV}}(x, y)$.

We first show that $\beta^*(A^*) = y(\mathrm{pre}(A^*))$. Suppose otherwise $\beta^*(A^*) < y(\mathrm{pre}(A^*))$. Let $R = \{r_{s,\sigma}\}$ be the reporting strategy that gives $y \to \beta^*$. We know that there exists some $s_0 \in \mathrm{pre}(A^*)$ with $y(s_0) > 0$ where $R$ does not convert all probability mass on $s_0$ into signals in $A^*$. Formally, we have $\sum_{\sigma \in A^*: (s_0,\sigma) \in E} r_{s_0,\sigma} < 1$. Consider any $\sigma_1, \sigma_2 \in \Sigma$ satisfying: $\sigma_1 \notin A^*$, $(s_0, \sigma_1) \in E$, $r_{s_0,\sigma_1} > 0$, $\sigma_2 \in A^*$, and $(s_0, \sigma_2) \in E$. We have $\alpha^*(\sigma_1) < \beta^*(\sigma_1)$ and $\alpha^*(\sigma_2) \geq \beta^*(\sigma_2)$. Now we discuss the following two cases and show there is a contradiction in both cases.

- If $\alpha^*(\sigma_2) > \beta^*(\sigma_2)$, then by moving

  $$\min\{r_{s_0,\sigma_1} y(s_0), \beta^*(\sigma_1) - \alpha^*(\sigma_1), \alpha^*(\sigma_2) - \beta^*(\sigma_2)\} > 0$$

  mass from $\sigma_1$ to $\sigma_2$, $y$ can report $\beta'$ such that $d_{\mathrm{TV}}(\alpha^*, \beta') < d_{\mathrm{TV}}(\alpha^*, \beta^*)$, a contradiction.
- If $\alpha^*(\sigma_2) = \beta^*(\sigma_2)$, then by moving

  $$\min\{r_{s_0,\sigma_1} y(s_0), (\beta^*(\sigma_1) - \alpha^*(\sigma_1))/2\} > 0$$

  mass from $\sigma_1$ to $\sigma_2$, $y$ can report $\beta'$ such that $d_{\mathrm{TV}}(\alpha^*, \beta^*) = d_{\mathrm{TV}}(\alpha^*, \beta')$. But now $\alpha^*(\sigma_2) - \beta'(\sigma_2) < 0$, and for any $\sigma \neq \sigma_2$, the sign of $\alpha^*(\sigma) - \beta'(\sigma)$ is the same as that of $\alpha^*(\sigma) - \beta^*(\sigma)$. So we have

  $$|\{\sigma \mid \alpha^*(\sigma) \geq \beta^*(\sigma)\}| > |\{\sigma \mid \alpha^*(\sigma) \geq \beta'(\sigma)\}|,$$

which contradicts the choice of $\beta^*$.

Now given that $y(\text{pre}(A^*)) = \beta^*(A^*)$, we have

$$\begin{aligned}
\text{MaxSep}(x, y) &= \max_A (x(\text{pre}(A)) - y(\text{pre}(A)) \\
&\geq x(\text{pre}(A^*)) - y(\text{pre}(A^*)) \\
&\geq \alpha^*(A^*) - y(\text{pre}(A^*)) && \text{(Proposition 6.5)} \\
&= \alpha^*(A^*) - \beta^*(A^*) \\
&= d_{\text{TV}}(\alpha, \beta) \\
&= d_{\text{DTV}}(x, y). && \square
\end{aligned}$$

*Proof of Proposition 6.2.* Let $A^* = \text{argmax}_A(x(\text{pre}(A)) - z(\text{pre}(A)))$. We have

$$\begin{aligned}
d_{\text{DTV}}(x, y) + d_{\text{DTV}}(y, z) &= \text{MaxSep}(x, y) + \text{MaxSep}(y, z) \\
&= \max_A (x(\text{pre}(A)) - y(\text{pre}(A))) + \max_A (y(\text{pre}(A)) - z(\text{pre}(A))) \\
&\geq (x(\text{pre}(A^*)) - y(\text{pre}(A^*))) + (y(\text{pre}(A^*)) - z(\text{pre}(A^*))) \\
&= x(\text{pre}(A^*)) - z(\text{pre}(A^*)) \\
&= \text{MaxSep}(x, z) \\
&= d_{\text{DTV}}(x, z). && \square
\end{aligned}$$

## 6.8 Omitted Proofs From Section 6.4

*Proof of Theorem 6.1.* Part (i) follows from Theorem 6.3.

For part (ii), suppose $d_{\text{DTV}}(g, b) = 0$. Let $s_g^i$ (resp. $s_b$) be a random variable that denotes the sample drawn from $g$ (resp. $b$) at time $i$. Abusing notation, for two random variables $X$ and $Y$, we write $d_{TV}(X, Y)$ for the $d_{TV}$ between the underlying distributions of $X$ and $Y$.

We show that given an adaptive / non-adaptive $\mathcal{R}_g$, there is an adaptive / non-adaptive $\mathcal{R}_b$, such that

$$d_{\text{TV}}((R_g^i(s_g^i))_{i\in[T]}, (R_b^i(s_b^i))_{i\in[T]}) = 0. \tag{6.1}$$

Because the good and bad distributions have identical distributions over the signal space, and this holds for all possible reporting strategies $\mathcal{R}_g$, part (ii) follows immediately.

Consider first non-adaptive reporting. Fix $\mathcal{R}_g = (R_g^1, \ldots, R_g^T)$ where $R_g^i = R_g$ for all $i$, let $\mathcal{R}_b = (R_b^1, \ldots, R_b^T)$, where

$$d_{\text{TV}}(R_g^i(s_g^i), R_b^i(s_b^i)) = 0.$$

The existence of such an $\mathcal{R}_b$ follows from the fact that $d_{\text{DTV}}(g, b) = 0$. Now since $R_g^i(s_g^i)$ and $R_b^i(s_b^i)$ are i.i.d., Equation (6.1) holds.

Now consider adaptive reporting. For any adaptive reporting strategy $\mathcal{R}_g$, we will construct an adaptive $\mathcal{R}_b$ inductively, such that for any $k$,

$$d_{\text{TV}}((R_g^i(s_g^i))_{i\in[k]}, (R_b^i(s_b^i))_{i\in[k]}) = 0.$$

For the base case when $k = 1$, observe that since $d_{\mathrm{DTV}}(g, b) = 0$, for any $R_g^1$, there exists $R_b^1$ such that

$$d_{\mathrm{TV}}(R_g^1(s_g^1), R_b^1(s_b^1)) = 0.$$

For the inductive case, suppose that $d_{\mathrm{TV}}((R_g^i(s_g^i))_{i\in[k]}, (R_b^i(s_b^i))_{i\in[k]}) = 0$. Given $(R_b^1, \ldots, R_b^k)$, we construct $R_b^{k+1}$ in the following way. Let $R_b^{k+1}$ be such that

$$R_b^{k+1}(s_b^{k+1} \mid \sigma^1, \ldots, \sigma^k) = R_g^{k+1}(s_g^{k+1} \mid \sigma^1, \ldots, \sigma^k),$$

for any $(\sigma^1, \ldots, \sigma^k)$. Now for any $(\sigma^1, \ldots, \sigma^{k+1})$,

$$\begin{aligned}
&\Pr[(R_b^1(s_b^1), \ldots, R_b^{k+1}(s_b^{k+1})) = (\sigma^1, \ldots, \sigma^{k+1})] \\
&= \Pr[(R_b^1(s_b^1), \ldots, R_b^k(s_b^k)) = (\sigma^1, \ldots, \sigma^k)] \cdot \Pr[R_b^{k+1}(s_b^{k+1} \mid \sigma^1, \ldots, \sigma^k) = \sigma^{k+1}] \\
&= \Pr[(R_g^1(s_g^1), \ldots, R_g^k(s_g^k)) = (\sigma^1, \ldots, \sigma^k)] \cdot \Pr[R_b^{k+1}(s_b^{k+1} \mid \sigma^1, \ldots, \sigma^k) = \sigma^{k+1}] \\
&\qquad\qquad \text{(induction hypothesis)} \\
&= \Pr[(R_g^1(s_g^1), \ldots, R_g^k(s_g^k)) = (\sigma^1, \ldots, \sigma^k)] \cdot \Pr[R_g^{k+1}(s_g^{k+1} \mid \sigma^1, \ldots, \sigma^k) = \sigma^{k+1}] \\
&\qquad\qquad \text{(construction of } R_b^{k+1}\text{)} \\
&= \Pr[(R_g^1(s_g^1), \ldots, R_g^{k+1}(s_g^{k+1})) = (\sigma^1, \ldots, \sigma^{k+1})].
\end{aligned}$$

In other words, we have

$$d_{\mathrm{TV}}((R_g^i(s_g^i))_{i\in[k+1]}, (R_b^i(s_b^i))_{i\in[k+1]}) = 0,$$

which concludes the inductive proof for Equation (6.1) in the adaptive case. □

*Proof of Theorem 6.2.* We reduce from Set Cover. More specifically, we use the following decision version of Set Cover: given ground set $X = [n]$, family of sets $\mathcal{F} = \{F_1, \ldots, F_m\}$ where $F_i \subseteq X$, and integer $k = m/2$, determine whether there are $k$ sets in $\mathcal{F}$ whose union is $X$. Note that it is without generality to set $k = m/2$, since given any Set Cover instance with an arbitrary $k$, we could always pad the instance by adding at most $m$ elements into $X$ and $m$ sets into $\mathcal{F}$, to obtain an equivalent new instance with $k' = m'/2$. Fixing a Set Cover instance, we construct $S$, $\Sigma$, $E$, $x$ and $y$ in the following way.

- $S = U \cup V$, where $U = \{u_1, \ldots, u_{n+1}\}$, $V = \{v_1, \ldots, v_{m+2}\}$, and $U \cap V = \emptyset$.
- $\Sigma = \{\sigma_1, \ldots, \sigma_{m+1}\}$.
- $x(u_i) = \frac{1}{2n}$ for $i \in [n]$, and $x(u_{n+1}) = \frac{1}{2}$.
- $y(v_i) = (1/2 + 1/(2n) - t)/m$ for $i \in [m]$, $y(v_{m+1}) = \frac{1}{2} - \frac{1}{2n}$, and $y(v_{m+2}) = t$, where $t \in [0, 1/2 + 1/(2n)]$ is a constant to be determined later.
- For $i \in [m]$ and $j \in F_i$, let $(u_j, \sigma_i) \in E$. Let $(u_{n+1}, \sigma_{m+1}) \in E$.
- For any $i \in [m]$, let $(v_i, \sigma_i) \in E$. For any $i \in [m+2]$, let $(v_i, \sigma_{m+1}) \in E$. For any $i \in [m+1]$, let $(v_{m+1}, \sigma_i) \in E$.
- $E$ contains only edges that mentioned above.

Now consider the problem of finding a set that separates $x$ from $y$ with a positive margin. First observe that such a set $A$ would never include $\sigma_{m+1}$, since $y(\mathrm{pre}(\{\sigma_{m+1}\})) = 1$. Our goal is to set $t$, such that iff $|A| \le k$ and $\mathrm{pre}(A) = \{u_1, \ldots, u_n\}$, $A$ separates $x$ from $y$ with a positive margin. Such an $A$ in the Set Cover instance would correspond to at most $k$ sets in $\mathcal{F}$ whose union cover $X$. Note that if $\sigma_{m+1} \notin A$,

$$y(\mathrm{pre}(A)) = \frac{1/2 + 1/(2n) - t}{m} \cdot |A| + \frac{1}{2} - \frac{1}{2n} \ge \frac{1}{2} - \frac{1}{2n}.$$

If $\mathrm{pre}(A)$ covers $\{u_1, \ldots, u_n\}$, then $x(\mathrm{pre}(A)) = \frac{1}{2}$. Otherwise, $x(\mathrm{pre}(A)) \le \frac{1}{2} - \frac{1}{2n} \le y(\mathrm{pre}(A))$. So if $\mathrm{pre}(A)$ does not cover $\{u_1, \ldots, u_n\}$, $A$ cannot be a separating set. We set $t$ such that $y(\mathrm{pre}(A)) = \frac{1}{2}$ if $|A| = k + 1 = (m+2)/2$. Such a $t$ always exists. Moreover, observe that such a value of $t$ guarantees that whenever $|A| \le k$, $y(\mathrm{pre}(A)) \le \frac{1}{2} - \frac{1}{\mathrm{poly}(m,n)}$. Now iff $|A| \le k$ and $A$ covers $\{u_1, \ldots, u_n\}$, $A$ separates $x$ from $y$ with a margin of $\frac{1}{\mathrm{poly}(m,n)}$. In other words, there is a separating set with a positive margin iff there are at most $k$ sets that cover $X$ in the Set Cover instance. Our NP-hardness result follows. □

*Proof of Theorem 6.3.* For the first bullet point, let $A^*$ be a set which separates $g$ from $b$ by a margin of $\varepsilon$. Consider the following policy: accept $(\sigma^1, \ldots, \sigma^T)$ iff

$$\frac{1}{T} \sum_{i \in [T]} \mathbb{I}[\sigma^i \in A^*] \ge g(\mathrm{pre}(A^*)) - \frac{1}{2}\varepsilon.$$

That is, the policy accepts the distribution iff $\bar{\alpha}(A^*) \ge g(\mathrm{pre}(A^*)) - \frac{1}{2}\varepsilon$, where $\bar{\alpha}$ is the empirical distribution of the reported signals. We now bound the probability of $g$ being accepted. Using some saturating reporting strategy $R_{A^*}$ for $A^*$ (Proposition 6.5), we have

$$s_g^i \in \mathrm{pre}(A^*) \iff R_{A^*}(s_g^i) \in A^*.$$

So by the Chernoff-Hoeffding bound, $f$ rejects $g$ with probability

$$\Pr\left[\frac{1}{T}\sum_i \mathbb{I}[s_g^i \in \mathrm{pre}(A^*)] - g(\mathrm{pre}(A^*)) < -\frac{1}{2}\varepsilon\right] \le \exp(-T\varepsilon^2/2) \le \delta.$$

On the other hand, by Proposition 6.5 for any reporting strategy $R_b$ of $b$,

$$\Pr[R_b(s_b^i) \in A^*] \le b(\mathrm{pre}(A^*)) \le g(\mathrm{pre}(A^*)) - \varepsilon.$$

So $f$ accepts $b$ with probability at most

$$\Pr\left[\frac{1}{T}\sum_i \mathbb{I}[s_b^i \in \mathrm{pre}(A^*)] - b(\mathrm{pre}(A^*)) \ge \frac{1}{2}\varepsilon\right] \le \exp(-T\varepsilon^2/2) \le \delta.$$

For the second bullet point, consider the following instance: $S = \Sigma = (s_1, s_2)$, $g(s_1) = \frac{1}{2} + \varepsilon$, $g(s_2) = \frac{1}{2} - \varepsilon$, $b(s_1) = b(s_2) = \frac{1}{2}$, and $E = \{(s_1, s_1), (s_2, s_2)\}$. In words, $s_1$ is a good sample/signal, and $s_2$ is a bad one. Agents must report the sample drawn as is. The good distribution draws good samples with slightly higher probability than the bad distribution. For this instance, distinguishing between $g$ and $b$ is exactly equivalent to distinguishing a coin with bias $\varepsilon$ with a fair coin. In the latter problem, it is well-known that $\Omega(1/\varepsilon^2)$ samples are required. □

*Proof of Theorem 6.4.* Consider the following policy which uses the policy in Theorem 6.3 as a building block. Let the policy in Theorem 6.3 be $f_{g,b}$ for good distribution $g$ and bad distribution $b$. Let $T_0 = 2\ln(k\ell/\delta)/\varepsilon^2$, where $\ell T_0 = T$. Given the $T$ reported signals $(\sigma^i)$, our policy $f$ proceeds in the following way:

- For each $i \in [k]$, $j \in [\ell]$, feed the $T_0$ signals

$$\sigma^{(j-1)T_0+1}, \ldots, \sigma^{jT_0}$$

  to policy $f_{g_i,b_j}$, and let the output be $o_{i,j} = f_{g_i,b_j}(\sigma^{(j-1)T_0+1}, \ldots, \sigma^{jT_0})$.
- $f$ outputs 1 iff

$$\bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} o_{i,j} = 1.$$

To see the correctness of the policy, observe that for each any $i, j$, with probability $1 - \frac{\delta}{k\ell}$, $f_{g_i,b_j}$ accepts $g_i$ and rejects $b_j$ given the signals fed in. Taking a union bound over all such $(i, j)$, with probability at least $1-\delta$, all these policies succeed simultaneously. Now for some good distribution $g_{i^*}$, as long as the above event happens, we have $o_{i^*,j} = 1$ for all $j \in [\ell]$, so

$$\bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} o_{i,j} \geq \prod_{j\in[\ell]} o_{i^*,j} = 1.$$

On the other hand, for some bad distribution $b_{j^*}$, we have $o_{i,j^*} = 0$ for any $i \in [k]$, and therefore

$$\bigvee_{i\in[k]} \bigwedge_{j\in[\ell]} o_{i,j} \leq \sum_i \prod_j o_{i,j} = 0. \qquad \square$$

## 6.9 Omitted Proofs From Section 6.5

*Proof of Proposition 6.3.* Consider $f'$ (resp. $f''$) which first applies the optimal non-adaptive (resp. adaptive) reporting strategy for $x$ to the original samples, and then applies $f$ to the transformed samples. Now the optimal reporting strategy for $x$ given policy $f'$ (or $f''$) is simply reporting the original sample received from $x$. The proposition follows. $\square$

*Proof of Proposition 6.4.* We show that $\mathrm{MaxSep}(x, y) = \min_{y\to y'} d_{\mathrm{TV}}(x, y')$, which implies the proposition given Proposition 6.1.

Consider the following flow network $G = (V, E', w)$:

- $V = S \cup \{s_s, s_t\}$, where $s_s$ is the source and $s_t$ is the sink.
- $E' = E \cup \{(s_s, s)\}_{s\in S} \cup \{(s, s_t)\}_{s\in S}$.
- $w(s_1, s_2) = \infty$ for any $(s_1, s_2) \in E$, $w(s_s, s) = b(s)$ for $s \in S$, and $w(s, s_t) = g(s)$ for $s \in S$.

See Figure 6.2 for illustration of an example network. Now observe that

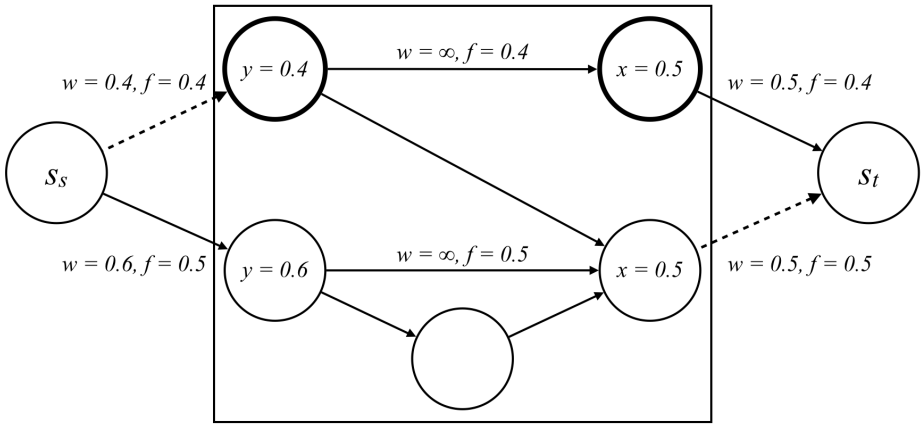


Figure 6.2: Illustration of Proposition 6.4. Vertices in the frame are from $S$, and the rest of the network is constructed as described in the proof. The dashed edges are saturated in the max flow. The boldface vertices are cut to $s_t$, and therefore constitute the prefix supporting the max separation.

- $1 - \mathrm{MaxSep}(x, y)$ is the $s_s$-$s_t$ min-cut of this network. This is because every set $A \subseteq S$ corresponds to a cut, where $S \setminus \mathrm{pre}(A)$ is cut to $s_s$ and $\mathrm{pre}(A)$ is cut to $s_t$. The value of $1 - (x(\mathrm{pre}(A)) - y(\mathrm{pre}(A)))$ is exactly the value of the cut. Similarly, any cut corresponds to a separating set. It follows that $\mathrm{MaxSep}(x, y)$ corresponds to the min-cut.
- $1 - \min_{y \to y'} d_{\mathrm{TV}}(x, y')$ is the $s_s$-$s_t$ max-flow of the network. This is because every $y'$ corresponds to a feasible flow in the network, whose capacity is

$$\sum_s \min\{x(s), y'(s)\} = 1 - d_{\mathrm{TV}}(x, y').$$

  Taking $\max$ over $y'$, we see that the max-flow has capacity

$$\max_{y \to y'} (1 - d_{\mathrm{TV}}(x, y')) = 1 - \min_{y \to y'} d_{\mathrm{TV}}(x, y').$$

Strong duality immediately gives the desired statement. □

*Proof of Corollary 6.1.* Run max-flow on the flow network constructed in the proof of Proposition 6.4, compute the min-cut on the residual network, and return the subset of $S$ on the same side as $s_s$. □

*Proof of Theorem 6.5.* Let the policy in Theorem 6.3 be the *truthful version* of $f_{g,b}$ for good distribution $g$ and bad distribution $b$.[3] Given the $T$ reported signals $(\sigma^i)$, our policy $f$ proceeds in the following way:

[3]The policy in Theorem 6.3 is itself truthful, but the construction here works even if it is not.

- For each $i \in [k]$, $j \in [\ell]$, feed all $T$ signals reported to policy $f_{g_{i,b_j}}$, and let the output be $o_{i,j} = f_{g_{i,b_j}}(\sigma^1, \ldots, \sigma^T)$.
- $f$ outputs 1 iff

$$\bigvee_{i \in [k]} \bigwedge_{j \in [\ell]} o_{i,j} = 1.$$

The rest of the proof is essentially the same as that of Theorem 6.4. □

Our policy against any adaptive bad distribution in Theorem 6.6 uses an efficient learner as a building block, which generalizes classical results for learning discrete distributions.

**Theorem 6.8** (Efficient Learner)**.** *Let $\rho = \rho(G)$ be the width of graph $G = (S, E)$. For any $x \in \Delta(S)$, $\varepsilon > 0$, $\delta > 0$, and $T = \frac{\rho \ln(1+n/\rho) \ln(1/\delta)}{2\varepsilon^2}$, for any valid reporting strategy that satisfies $(s^i, \sigma^i) \in E$, with probability at least $1 - \delta$, $d_{\mathrm{DTV}}(\bar{\alpha}, x) \le \varepsilon$, where $\bar{\alpha}$ is the empirical distribution given by the reports $(\sigma^i)_i$, i.e., $\bar{\alpha}(s) = \frac{\sum_i \mathbb{I}[\sigma^i = s]}{T}$.*

The following well-known fact about the width is used in the analysis of our learner:

**Theorem 6.9** (Dilworth's Theorem)**.** *A* chain *in a partially ordered set $G = (S, E)$ is an ordered set $C = (c_1, \ldots, c_\ell)$, where $c_i \in S$ for $i \in [\ell]$ and $(c_i, c_{i+1}) \in E$ for any $i \in [\ell - 1]$. Dilworth's Theorem states that for any partially ordered set $G = (S, E)$, the width of $\rho(G)$ is equal to the minimum number of chains whose union covers $S$.*

*Proof of Theorem 6.8.* We show that $\mathrm{MaxSep}(\bar{\alpha}, x) \le \varepsilon$ w.p. $1 - \delta$. More specifically, if for all $A$ where $A = \mathrm{pre}(A)$, $\bar{\alpha}(A) - x(A) \le \varepsilon$, then duality gives immediately that $d_{\mathrm{DTV}}(\bar{\alpha}, x) \le \varepsilon$. We will show that this happens with probability $1 - \delta$.

Let $\bar{x}$ be the empirical distribution of $(s^i)_i$. Fix $A \subseteq S$ where $A = \mathrm{pre}(A)$. Observe that $\bar{x}(A) \ge \bar{\alpha}(A)$, so $x(A) = \mathbb{E}[\bar{x}(A)] \ge \mathbb{E}[\bar{\alpha}(A)]$. The Chernoff bound gives

$$\Pr[\bar{\alpha}(A) \ge x(A) + \varepsilon] \le \exp(-2T\varepsilon^2) \le \frac{\delta}{(1 + n/\rho)^\rho}.$$

We only need to show that the number of different sets $A$ where $A = \mathrm{pre}(A)$ is at most $(1 + n/\rho)^\rho$. We call such sets prefixes of graph $(S, E)$. Dilworth's Theorem (Theorem 6.9) states that the width $\rho$ of $(S, E)$ is equal to the minimum number of chains whose union covers $S$. Let $\mathcal{C} = \{C_k\}_{k \in [\rho]}$ be such a covering family, where for any $k$, $C_k = (s_{k,1}, \ldots, S_{k,\ell_k})$ is a chain (i.e., $(s_{k,i}, s_{k,i+1}) \in E$ for $i \in [\ell_k - 1]$. For any prefix $A$, let $p_k(A) = |A \cap C_k|$. Observe that if two prefixes $A_1$ and $A_2$ are distinct, then there is some $k \in [\rho]$ such that $p_k(A_1) \ne p_k(A_2)$. On the other hand, consider vector $(p_1(A), \ldots, p_\rho(A))$. The number of possible values of this vector is $\prod_k (\ell_k + 1) \le (1 + n/\rho)^\rho$, which is an upper bound of the number of different prefixes. Taking union bound over all these prefixes, we have

$$\Pr[\forall A \text{ where } A = \mathrm{pre}(A), \bar{\alpha}(A) \ge x(A) + \varepsilon] \le \frac{\delta}{(1 + n/\rho)^\rho} \cdot (1 + n/\rho)^\rho = \delta.$$

The theorem follows. □

Given the efficient learner constructed above, we are ready to prove Theorem 6.6.

*Proof of Theorem 6.6.* Consider the following policy: compute the empirical distribution $\bar{\alpha}$ of the reported signals. Accept iff $d_{\mathrm{DTV}}(g, \bar{\alpha}) < \frac{1}{2}\varepsilon$. Note that since $g$ is known, $d_{\mathrm{DTV}}(g, \bar{\alpha})$ can be computed in polynomial time using the algorithm in Corollary 6.1.

We first show that $p_{\mathrm{non}}(f, g, T) \geq 1 - \delta)$. In particular, we show that if $g$ reports truthfully, then with probability $1 - \delta$, $d_{\mathrm{DTV}}(g, \bar{g}) < 1 - \frac{1}{2}\varepsilon$. The argument is similar to that in the proof of Theorem 6.8. For any $A \subseteq S$ where $A = \mathrm{pre}(A)$, the Chernoff bound implies

$$\Pr[g(A) - \bar{g}(A) \geq \varepsilon/2] \leq \frac{\delta}{(1 + n/\rho)^{\rho}}.$$

Since there are at most $(1 + n/\rho)^{\rho}$ such sets, from a simple union bound, with probability $1 - \delta$, $d_{\mathrm{DTV}}(g, \bar{g}) = \mathrm{MaxSep}(g, \bar{g}) \leq \frac{1}{2}\varepsilon$.

Now we show that $p_{\mathrm{ada}}(f, b, T) \leq \delta$ for any $b$ where $d_{\mathrm{DTV}}(g, b) \geq \varepsilon$. No matter what adaptive reporting strategy $b$ uses, the signals reported by $b$ must satisfy $(s_b^i, \sigma_b^i) \in E$ for all $i$. By Theorem 6.8, with probability $1 - \delta$, the empirical distribution $\bar{\alpha}$ satisfies $d_{\mathrm{DTV}}(\bar{\alpha}, b) \leq \frac{1}{2}\varepsilon$. Now since $d_{\mathrm{DTV}}$ satisfies the triangle inequality (Proposition 6.2),

$$d_{\mathrm{DTV}}(g, \bar{\alpha}) \geq d_{\mathrm{DTV}}(g, b) - d_{\mathrm{DTV}}(\bar{\alpha}, b) \geq \varepsilon - \frac{1}{2}\varepsilon = \frac{1}{2}\varepsilon.$$

Whenever this happens, $b$ is rejected by $f$, which means $p_{\mathrm{ada}}(f, b, T) \leq \delta$. □

*Proof of Theorem 6.7.* We use the algorithm by Valiant and Valiant [99] for testing identity of discrete distributions as a building block. Given a distribution $x \in \Delta([n])$, with $T = O\left(\frac{\sqrt{n}\ln(1/\delta)}{\varepsilon^2}\right)$ samples to an unknown distribution $y$, their algorithm distinguishes between the following two cases: (1) $y = x$ and (2) $d_{\mathrm{TV}}(x, y) \geq \varepsilon$. Our policy for non-adaptive reporting is simply running the algorithm by Valiant and Valiant on the good distribution $g$ and the signals reported $(\sigma^i)_i$.

The good distribution $g$, in order to be accepted with high probability, simply reports truthfully. The distribution of signals of $g$ is therefore exactly $g$, which with probability $1 - \delta$ passes the test.

As for the bad distribution, observe that any non-adaptive reporting strategy $\mathcal{R}_b = (R_b, \ldots, R_b)$ induces a distribution $\alpha_b$ of signals reported, where $b \to_{R_b} \alpha_b$. No matter how $b$ reports, because $d_{\mathrm{DTV}}(g, b) \geq \varepsilon$, we always have $d_{\mathrm{TV}}(g, \alpha_b) \geq \varepsilon$, in which case $\alpha_b$ fails the test with probability at least $1 - \delta$. □

# Chapter 7

# Classification with Few Samples through Self-Selection

## 7.1 Introduction

In previous chapters, we have discussed sample-based classification in settings where samples can either be strategically selected, or strategically transformed by self-interested agents. In this chapter, we investigate a closely related setting, where samples are not exogenous, but endogenous. That is, the agent can choose how many samples they want to generate. This corresponds to classification based on the outcomes of *tests*.

In a narrow sense, tests can take the form of exams, with numerical scores as *outcomes*. For example, a course often has one or more midterm exams and one final exam, and the intructor uses the outcomes of these exams to decide the final grades of (i.e., to *classify*) students. More generally, a test can be any activity that takes a certain amount of effort and produces a verifiable outcome. Examples include job interviews, research paper submissions, etc. These outcomes, presumably correlated with the true skills of the test takers (henceforth the *agents*), are then used by a *principal* to classify them — the collective feedback from different interviewers determines whether the interviewee gets the job, and the list of papers one has published strongly affects one's future opportunities as a researcher.

An agent's performance on tests is inevitably random — on any given day, a capable student may not perform well due to being tired or sick, due to bad luck in which questions were selected, or for reasons that we cannot identify. For this reason, a principal generally is willing to take into consideration multiple test outcomes when making decisions. It then matters how these tests are offered to agents. Oversimplifying, there are two ways of offering tests: *mandatory* tests and *optional* tests. With mandatory tests, the principal decides which tests each agent should take and/or how many times they should take them, as well as which (combinations of) outcomes an agent needs to have in order to be classified into a certain category. A straightforward example of mandatory tests is students taking exams in school, where typically all students are required to take all exams in a course, whose outcomes together determine the final grade of the student. On the other hand, with optional tests, the principal decides the latter (i.e., which outcomes suffice for

classification into a certain category) but not the former (i.e., which and/or how many tests each agent should take). One example is (an oversimplified version of) the academic job market, where agents' publication records determine whether they are invited for an onsite interview, but agents can decide how often to put in the effort to prepare a new paper for submission to a conference or journal (i.e., to "take" and optional "test"). At first glance, it may appear that mandatory tests allow the principal tighter control over the classification process, and therefore would benefit the principal more than optional tests. As a consequence, the principal should enforce mandatory tests whenever possible (or economically feasible). However, the above intuitive reasoning does not appear to be fully backed by evidence from reality: optional tests continue to be implemented in high-stakes classification tasks such as US college admissions.[1] This raises the following question:

*Are there any advantages of optional tests for classification over mandatory ones?*

On top of that, in many other scenarios mandatory tests are simply unrealistic, and the principal has to rely on optional tests for decision-making — for example, it is impossible for an academic hiring committee to *require* job applicants to submit their work to certain conferences in a prescribed way, e.g., one paper to NeurIPS'20 and one paper to ICML'21. In such cases, the principal would still like the classification process to be as accurate and efficient as possible. This leads us to the following question:

*How can one design a classification process with optional tests in the most accurate and/or efficient way?*

**Our results.** We give somewhat surprising answers to the above two questions in the case of binary classification of binary agents (elaborated below): we characterize the optimal design of a classification process with optional tests, and based on this show that classification with optional tests can be arbitrarily more efficient than optimal classification with mandatory tests for the same task.

To be more specific, we consider a setting where a principal either accepts or rejects agents based on the set of test outcomes that they get. Before tests are taken, the principal commits to a policy, which consists of all sets of outcomes that lead to acceptance. Each agent is modeled by a distribution over the space of possible outcomes, corresponding to how the agent tends to perform in a test. When an agent takes a test, he pays a cost and receives an independent sample from his distribution as the outcome. Agents can always choose between taking another test, and stopping. They maximize their expected utility, which is the value of acceptance if the principal's policy accepts the set of outcomes they have and $0$ otherwise, minus the total cost of tests taken.

We focus on the case where agents can be either "good" or "bad" (corresponding to two different distributions over test outcomes), and the principal's goal is to accept good agents and reject bad ones. We first characterize the optimal strategy of an agent in response to the principal's policy. Fixing the principal's policy, each agent faces a Markov Decision Process (MDP), where the state is the set of test outcomes that he has collected. In general, at any state of the MDP, the agent can always choose between taking another test and stopping, and the optimal strategy could be any function mapping each combination of outcomes to one of the two actions. Our first key

[1] As discussed in Chapter 4, applicants may take SAT and/or ACT tests, among others, as many times as they want.

observation is that without loss of generality, the agent's optimal strategy is either to keep taking tests until acceptance, or to leave immediately without taking any test. This is because intuitively, after taking some tests, the agent must have received some outcomes, which makes his situation at least as good as when he started in terms of the expected number of *future* tests he needs to take in order to be accepted; the cost of the tests already taken is sunk. So, if an agent ever chooses to start taking tests, he must be willing to keep taking tests until acceptance, since the cost of past tests should not affect his decision. This is making two assumptions: (1) the agent can choose not to submit some of the test outcomes, and (2) the agent already knows his own type (good or bad) at the beginning, and hence is not learning about himself from the test outcomes.

With agents' optimal strategy characterized, we consider the principal's problem, i.e., the design of her classification policy. We first study the case where the principal controls the cost of a test, by, for example, charging a registration fee. We show that in this case, as long as the good and bad agents have different distributions (which can be arbitrarily close to each other), the principal can always achieve perfect accuracy, meaning good agents are accepted with probability $1$, and bad ones are rejected with probability $1$. The key technique is to choose the policy so that agents self-select into (not) taking tests. Moreover, among perfectly accurate policies, we characterize the one with stochastically dominant efficiency in terms of the number of tests a good agent needs to take in order to be accepted. We show that quite surprisingly, under this policy, no agent ever has to take more than $2$ tests. One may contrast this with the mandatory tests case, where the principal directly observes as many samples as she wants from agents' distributions — there, in order to classify correctly with probability $2/3$, the number of tests required can be arbitrarily large, as the distance between the good and bad distributions diminishes.

We then proceed to the case where the cost per test is fixed externally. With discrete outcomes, we show that perfect accuracy in general is no longer possible. We then consider the special case with continuous outcome distributions, or equivalently, where outcomes are associated with rich noise that is effectively continuous. We show that in a continuous world, with different good and bad distributions, perfect accuracy is again always possible. Moreover, we construct a perfectly accurate policy under which the maximum number of tests a good agent needs to take is $\lfloor 1/c \rfloor + 1$ where $c$ is the fixed cost per test, and show this is essentially best possible for perfectly accurate policies. We also provide evidence that the above bound cannot be significantly improved even if we consider the expected number of tests.

## 7.2 Preliminaries

In this section, we formally define the problem of test-based classification.

**Agents, tests, and outcomes.** Each agent is modeled by a distribution $D$ over a space $O$ of possible test outcomes (e.g., integers between $0$ and $100$, corresponding to numerical scores). Agents can choose to take as many tests as they want. When an agent with distribution $D$ takes a test, he receives an outcome drawn from $D$ independently of past test outcomes, and can then choose to continue taking tests, or to stop. The outcomes of all the tests taken (which form a multiset whose elements are from the outcome space $O$) will then be used by the principal for classification.

**The principal and the policy.** The principal, before agents decide whether or not to take tests, announces a policy $\mathcal{P}$ for classification. The policy in general is a collection of multisets, each of which consists of certain test outcomes from the outcome space $O$. For an agent with outcomes $S$, the policy $\mathcal{P}$ accepts the agent iff there is a multiset $T \in \mathcal{P}$ such that $T \subseteq S$ (i.e., the multiplicity of any element in $T$ is no larger than that of the same element in $S$). In other words, the policy $\mathcal{P}$ provides a collection of options to agents, each of which is a multiset $T$ of outcomes. An agent is accepted iff his multiset of test outcomes contains any of these options as a subset. A simple and natural example is when $O$ is the set of integers between $0$ and $100$, and $\mathcal{P}$ contains a number of singleton multisets, each of which is an integer between $60$ and $100$, i.e.,

$$\mathcal{P} = \{\{i\} \mid 60 \leq i \leq 100\}.$$

This corresponds to the case where agents can repeatedly take exams, and are accepted (i.e., pass) iff they ever get a score of at least $60$. Another example would be

$$\mathcal{P} = \{\{i\} \mid 60 \leq i \leq 100\} \cup \{\{i, j\} \mid 50 \leq i, j < 60\},$$

which is the same policy as before, except it now also suffices to score at least 50 *twice*.

**How rational agents act in response to a policy.** Fixing a policy $\mathcal{P}$, each agent faces an MDP, where the goal is to maximize his expected utility. Below we describe this MDP. Without loss of generality, being accepted gives agents value $1$, and each test has a cost of $0 \leq c \leq 1$ (otherwise agents would never want to take any test). The states of the MDP, denoted $\mathcal{S}$, are all multisets over the outcome space $O$, corresponding to the set of outcomes the agent has collected so far. Initially, the state of the agent is the empty set $\emptyset$. At any state $S \in \mathcal{S}$, the agent can choose between two actions, taking another test ($\mathsf{T}$) or leaving ($\mathsf{L}$). If the agent chooses $\mathsf{T}$, he pays cost $c$ (i.e., receives reward $-c$), and transitions to a new state $S \cup \{o\}$ (note that this is a union of two multisets, where the multiplicity of any element in the union is the sum of those of the same element in the two operands), where $o \sim D$ is a random outcome drawn from $D$. If the agent chooses $\mathsf{L}$, he receives reward $1$ if his current multiset of outcomes $S$ is accepted by the policy $\mathcal{P}$, and $0$ otherwise; in either case, the MDP terminates immediately. Throughout the paper, we assume agents are perfectly rational and always play the utility-maximizing action. For simplicity, we assume agents always break ties in favor of leaving, i.e., when the two actions result in equal expected utility, they always play $\mathsf{L}$. Our results still hold (with minor modifications) even if agents break ties adversarially.

**The principal's goals.** We consider two goals of the principal, accuracy and efficiency. We focus on the case where agents are either good (with distribution $G$) or bad (with distribution $B$), and the principal aims to accept as many good agents as possible, and reject as many bad ones as possible. The specific definition of accuracy is immaterial — as we will show, the principal can always achieve perfect accuracy (i.e., good agents are always accepted, and bad ones always rejected) as long as $G$ and $B$ are not identical. Given perfect accuracy, the principal may further hope to implement the classification in an efficient way, where agents take as few tests as possible. We consider

two types of efficiency measures, the expected number of tests and the worst-case number of tests. The goal is to design perfectly accurate policies which (approximately) minimize either/both of these two measures (though in Section 7.4.3 we do consider how to minimize expected cost in our model under the constraint of perfect accuracy).

**Control over the costs.** In some scenarios, the cost of a test is controlled by the principal (e.g., when the dominant part of the cost is a registration fee set by the principal), while in others it is fixed externally (e.g., when the dominant part of the cost is time invested in traveling to the test site). We consider both cases in this paper. The flexible-cost case allows the principal refined control of the classification procedure, which, as we will show, implies more efficient policies in general.

## 7.3 Agents' Optimal Strategy: Self-Selection

We first characterize agents' best response to a policy, which effectively makes their decision space binary, and greatly simplifies the principal's problem.

**Lemma 7.1.** *Fixing a policy $\mathcal{P}$ and a cost per test $c$, the optimal expected reward of any agent is achieved by one of the following two strategies:*

- *Take no test (i.e., play $\mathsf{L}$ immediately) and leave with reward $0$.*
- *Keep taking tests (i.e., playing $\mathsf{T}$) until the set of outcomes collected is accepted by $\mathcal{P}$, and then play $\mathsf{L}$.*

*Moreover, the optimal strategy is unique iff the above two strategies result in strictly different expected rewards.*

The proof of Lemma 7.1, as well as all other proofs, is deferred to the appendix. Again, the intuition is that if an agent ever wants to start taking tests, then after taking some tests, he will be in at least as favorable a position as at the beginning in terms of tests passed, and it was worth it to start then, so it must certainly be worth it to continue now (the cost of previous tests is sunk, and therefore irrelevant). One important implication is that, depending on the policy, the cost per test, and the agent's distribution, each agent either does not attempt to get accepted at all, or keeps trying and eventually gets accepted with probability $1$. This indicates that, when provided the right incentives, self-selecting agents may perform the classification for the principal in a perfectly accurate way.

More specifically, for any policy $\mathcal{P}$ and distribution $D$ over the outcome space $O$, let $T(\mathcal{P}, D)$ denote the (random) number of tests an agent with distribution $D$ needs to take in order to be accepted by $\mathcal{P}$, i.e.,

$$T(\mathcal{P}, D) = \min\{t \mid \mathcal{P} \text{ accepts } \{o_1, \ldots, o_t\}\},$$

where $\{o_t\}_{t\geq 1}$ are iid draws from $D$. We have the following claim.

**Lemma 7.2.** *Fix a policy $\mathcal{P}$ and a cost per test $c$. An agent with distribution $D$ will always keep taking tests until acceptance if*

$$c \cdot \mathbb{E}_{\{o_t\}\sim D^{\mathbb{Z}_+}}[T(\mathcal{P}, D)] < 1,$$

*and leave immediately otherwise.*

In the rest of the chapter, we will heavily exploit Lemma 7.2.

## 7.4 The Flexible-Cost Case

We begin our investigation with the case where the cost of a test is set by the principal, which turns out to be simpler. For simplicity, we assume the outcome space $O = [k] = \{1, \ldots, k\}$ for some integer $k > 0$. For any distribution $D$ over $O$ (which can be either $G$ or $B$), for any $S \subseteq O$, let $D(S) = \Pr_{o \sim D}[o \in S]$. As a shorthand, for any $o \in O$, let $D(o) = D(\{o\})$. All the results in this section can be easily generalized to arbitrary outcome spaces.[2]

### 7.4.1 Memoryless Policies Suffice for Accurate Classification

We first consider the possibility of accurate classification. In particular, for reasons that will be clear momentarily, we are interested in policy-cost pairs that achieve perfect accuracy, as defined below.

**Definition 7.1** (Perfect Accuracy)**.** A policy-cost pair $(\mathcal{P}, c)$ is *perfectly accurate* for a good distribution $G$ and a bad distribution $B$ if the optimal strategies for good agents and bad agents respectively are to keep taking tests until acceptance and to leave immediately.

As a corollary of Lemma 7.2, a pair $(\mathcal{P}, c)$ is perfectly accurate iff

$$c \cdot \mathbb{E}[T(\mathcal{P}, G)] < 1 \leq c \cdot \mathbb{E}[T(\mathcal{P}, B)].$$

When the cost per test is controlled by the principal, we are further interested in policies that are perfectly implementable.

**Definition 7.2** (Perfect Implementability)**.** A policy $\mathcal{P}$ is *perfectly implementable* for a good distribution $G$ and a bad distribution $B$ if there exists a cost per test $c$, such that the policy-cost pair $(\mathcal{P}, c)$ achieves perfect accuracy.

Given Lemma 7.2, we immediately have the following necessary and sufficient condition for perfect implementability.

**Lemma 7.3.** *Fix a good distribution $G$ and a bad distribution $B$. A policy $\mathcal{P}$ is perfectly implementable iff*

$$\mathbb{E}[T(\mathcal{P}, G)] < \mathbb{E}[T(\mathcal{P}, B)].$$

Based on the above characterization, we show that as long as the good agents' distribution $G$ is different from the bad agents' distribution $B$, there always exists a perfectly implementable policy which consists of only singleton sets of outcomes. In other words, the policy is memoryless, in that a new test outcome either immediately makes the agent accepted, or will be entirely ignored.

[2]For example, when $O$ is an infinite, possibly continuous space (e.g., $O = \mathbb{R}$), one can discretize $O$ into a finite number (which may depend on the desired precision) of regions such that the good and bad distributions after discretization are arbitrarily close to the respective original distributions.

**Theorem 7.1.** *For any good distribution $G$ and bad distribution $B$ over the outcome space $O = [k]$ where $G \neq B$, there exists a set of outcomes $P \subseteq O$ such that the policy*

$$\mathcal{P} = \{\{o\} \mid o \in P\}$$

*is perfectly implementable.*

We remark that such memoryless policies are widely deployed in practice, where the most common form is to set a threshold and accept an agent iff the highest score he ever gets passes that threshold. However, as we will show later, this is not the most efficient form of perfectly accurate policies.

### 7.4.2 Policy with Stochastically Dominant Efficiency

We now proceed to efficient policies. Below we characterize the perfectly implementable policy $\mathcal{P}$ with stochastically dominant efficiency for any good distribution $G$ and bad distribution $B$. The number of tests required for a good agent to be accepted under this policy, $T(\mathcal{P}, G)$, stochastically dominates the same number, $T(\mathcal{P}', G)$, of any other perfectly implementable policy $\mathcal{P}'$. Furthermore, under this policy, any good agent is guaranteed to be accepted after taking at most 2 tests. As a result, this policy achieves the optimal expected number of tests, the optimal worst-case number of tests (which is 2), and optimality with respect to almost any reasonable measure of efficiency.

**Theorem 7.2.** *Let $P \subseteq O = [k]$ be a set of outcomes such that*

$$P \in \operatorname{argmax}_{S \subseteq O : G(S) > B(S)} G(S).$$

*The policy*

$$\mathcal{P} = \{\{o\} \mid o \in P\} \cup \{\{o_1, o_2\} \mid o_1, o_2 \in O\}$$

*is perfectly implementable, and stochastically dominates any other perfectly implementable policy $\mathcal{P}'$, in the sense that for any $t \in \mathbb{Z}_+$,*

$$\Pr[T(\mathcal{P}, G) \leq t] \geq \Pr[T(\mathcal{P}', G) \leq t].$$

The policy $\mathcal{P}$ constructed in Theorem 7.2 accepts any set of outcomes which either contains some outcome in $P \subseteq O$, or has cardinality at least 2. In other words, $\mathcal{P}$ accepts an agent if the first outcome he receives is in $P$, or he ever takes 2 tests. One may contrast Theorem 7.2 with the setting where the principal, rather than the agent himself, chooses the number of tests each agent needs to take. Suppose, rather than deploying a policy and letting agents themselves choose whether or not to take tests, we directly observe iid samples from an unknown distribution $D$, which can be either $G$ or $B$ — this corresponds to the case where we simply ask each agent to take as many tests as we want. There, how many samples one needs to observe in order to tell with confidence whether $D$ is $G$ or $B$ depends on the total variation distance between $G$ and $B$, defined below.

**Definition 7.3** (Total Variation Distance)**.** The total variation distance $d_{\mathrm{TV}}(D_1, D_2)$ between two distributions $D_1$ and $D_2$ over $O$ is defined as

$$d_{\mathrm{TV}}(D_1, D_2) = \sup_{S \subseteq O} (D_1(S) - D_2(S)).$$

Observe that $G \neq B$ iff $d_{\mathrm{TV}}(G, B) > 0$. It is folklore that in order to identify $D$ with probability at least $2/3$, one needs $\Omega(d_{\mathrm{TV}}(G, B)^{-2})$ iid samples from $D$. Moreover, it is easy to see that as long as the supports of $G$ and $B$ overlap, one can never be completely sure with any finite number of samples. Theorem 7.2, on the other hand, essentially says that whenever $d_{\mathrm{TV}}(G, B) > 0$, the principal never needs to observe more than 2 samples in order to distinguish $G$ and $B$, and good agents never need to take more than 2 tests. In other words, by incentivizing self-selection, the principal is able to reduce the number of tests required dramatically, from $\Omega(d_{\mathrm{TV}}(G, B)^{-2})$ to 2, and at the same time improve the accuracy to 1. Perhaps even more surprisingly, this is done by giving agents more freedom to choose the number of tests they take. Of course, this is feasible only because agents themselves know their distribution at the outset; if nobody knows the distribution, $\Omega(d_{\mathrm{TV}}(G, B)^{-2})$ tests would still be required. This partially explains the practical success of classification with optional tests: they can be arbitrarily more efficient than mandatory tests enforced by the principal, especially when good and bad agents' distributions are closer to each other and therefore are harder to distinguish.

### 7.4.3 Cost Efficiency of Policies

While the policy in Theorem 7.2 is efficient in terms of the *number of tests*, it could impose a total expected *cost* on good agents that is quite close to the benefit of being accepted. Depending on the circumstances, cost efficiency may be considered more important, and indeed, often the classification procedure can be implemented in much less costly ways, intuitively for the following reasons. First, when $G$ and $B$ are hard to distinguish, it is natural that good agents need to spend significant effort in order to distinguish themselves from bad ones. But, in many real-world scenarios, (most) good agents are considerably different from (most) bad ones. In such cases, good agents pay much less cost, since the principal only needs to make bad agents marginally unwilling to take tests. Second, there can be a tradeoff between efficiency (i.e., the number of tests taken) and cost. If the principal is willing to make good agents take more than 2 tests, then she can design a more selective policy (i.e., making it hard to pass) that creates a sharper separation between good and bad agents, and set a lower cost per test to achieve perfect accuracy. Below we formalize this intuition, and characterize the optimal cost efficiency possible, subject to perfect accuracy, for memoryless policies.

**Theorem 7.3.** *Fix any good distribution $G$ and bad distribution $B$ over the outcome space $O = [k]$ where $G \neq B$. There exists a memoryless policy*

$$\mathcal{P} = \{\{o\} \mid o \in P\}$$

*for some $P \subseteq O$, and a cost per test $c$, such that $(\mathcal{P}, c)$ is perfectly accurate, and the expected total cost paid by good agents is*

$$c \cdot \mathbb{E}[T(\mathcal{P}, G)] = \min_{o \in O} B(o)/G(o).$$

*Moreover, no policy-cost pair $(\mathcal{P}', c')$ satisfies (1) $\mathcal{P}'$ is memoryless, and (2) the expected total cost paid by good agents is*

$$c' \cdot \mathbb{E}[T(\mathcal{P}', G)] < \min_{o \in O} B(o)/G(o).$$

The above theorem says that the optimal cost efficiency achievable by memoryless policies is determined by the minimum ratio between $B$ and $G$ over the test outcome space. We also remark that cost efficiency directly implies robustness against bad agents who value acceptance more than good agents.[3] Fixing any good distribution $G$ and bad distribution $B$, when good agents have value 1 and bad agents have value $v \geq 1$ for acceptance, there exists a perfectly implementable policy iff the optimal cost efficiency achievable when all agents have value 1 is better than $v^{-1}$, i.e., there exists a policy-cost pair $(\mathcal{P}, c)$ such that

$$c \cdot \mathbb{E}[T(\mathcal{P}, G)] < v^{-1} \quad \text{and} \quad c \cdot \mathbb{E}[T(\mathcal{P}, B)] \geq 1.$$

In fact, given such a cost efficient pair $(\mathcal{P}, c)$, $(\mathcal{P}, v \cdot c)$ is a perfectly accurate pair when bad agents have value $v \geq 1$ for acceptance.

## 7.5 The Fixed-Cost Case

Now we proceed to the more challenging setting where the cost per test $c$ is fixed externally. We show that in such cases, perfect accuracy in general requires stronger conditions on the good and bad distributions. However, as we argue below, these conditions are still rather reasonable for practical purposes.

### 7.5.1 Accurate Classification Requires Continuous Information

When the cost per test is set by the principal, Theorem 7.1 states that perfect accuracy can be achieved by some policy-cost pair as long as the good and bad distributions are different. However, this is not true when the cost $0 < c < 1$ is fixed, as illustrated in the following example.

**Example 7.1.** Suppose the cost per test is fixed at $c = 0.9$. The outcome space $O = \{1, 2\}$, the good distribution $G$ assigns probability $G(1) = G(2) = 0.5$, and the bad distribution $B$ assigns $B(1) = 0$ and $B(2) = 1$. Suppose there is a policy $\mathcal{P}$ such that $(\mathcal{P}, c)$ is perfectly accurate. Then, in order for good agents to take tests, by Lemma 7.2,

$$\mathbb{E}[T(\mathcal{P}, G)] < 10/9,$$

and since $T(\mathcal{P}, G)$ is distributed over $\mathbb{Z}_+$, elementary calculation gives

$$\Pr[T(\mathcal{P}, G) = 1] > 8/9.$$

As a result, it must be the case that $\{1\} \in \mathcal{P}$ and $\{2\} \in \mathcal{P}$ simultaneously. However, this implies

$$\Pr[T(\mathcal{P}, B) = 1] = 1 \implies \mathbb{E}[T(\mathcal{P}, B)] < 10/9.$$

So bad agents will also take tests and get accepted under $(\mathcal{P}, c)$, a contradiction. In other words, no policy $\mathcal{P}$ exists such that $(\mathcal{P}, c)$ is perfectly accurate.

[3]The case where good agents value acceptance more is no harder than the case where all agents have the same value for acceptance.

The above example shows that perfect accuracy cannot be achieved with an infeasibly high cost per test, even if the outcome space is extremely simple (i.e., binary) and the good and bad distributions are clearly different. Nevertheless, the impossibility of perfect accuracy comes almost solely from the discreteness in the outcomes — intuitively, accepting only one of the two outcomes does not provide enough motivation for good agents to take tests, while accepting both provides too much motivation, so that every agent wants to take tests regardless of his distribution.

Real-world tests, however, are often intrinsically (approximately) continuous. In a narrow sense, test outcomes, in the form of numerical scores, usually range from $0$ to $100$, where presumably an agent can get any integer score in between with positive probability. As argued above, in a broader sense, a test could be any activity which takes a certain effort and produces a verifiable outcome. Besides numerical test scores, such an outcome could take the form of a course project, a research paper, or an oral presentation. These outcomes are essentially continuous, in the sense that, for example, no two oral presentations are exactly the same, even if they are given by the same presenter using the same slides. Even for relatively discrete outcome spaces, an outcome is often accompanied by arbitrarily rich noise, which makes outcomes effectively continuous.[4] For example, in a simplistic model, a paper submitted to a conference can be either accepted or rejected, so one could argue the outcome of such a submission is binary. However, it is extremely unlikely that two different papers (as PDF files) share the same hash value, which can effectively be viewed as continuous noise that we can add to the outcome, thereby making the outcome space continuous. This (hash value) part of the enriched outcome may not be correlated with the type of the agent, but that will not matter for our purposes.

Based on the above observations, in the rest of this section, we assume the outcome space $O$, as well as the good distribution $G$ and the bad distribution $B$, is continuous. This could model continuity in the outcome distribution itself, or noise, or the two aspects in combination. More specifically, without loss of generality, we assume $O = [0, k]$ for some positive integer $k \in \mathbb{Z}_+$, and the good distribution $G$ (resp. the bad distribution $B$) is constant when restricted to the interval $[i-1, i]$ for any $i \in [k] = \{1, \ldots, k\}$. We call such distributions piecewise constant.[5] One way to interpret this is that there are $k$ possible outcomes. A good (resp. bad) agent receives the $i$-th outcome with probability $G([i-1, i])$ (resp. $B([i-1, i])$). Moreover, there is continuous noise $x$ independent of the outcome and the agent type, uniformly distributed over $[0, 1]$, so the final combination of the outcome and the noise, $i - x$, has distribution $G$ (resp. $B$). As a shorthand, for any $i \in [k]$, let $G(i) = G([i-1, i])$, and $B(i) = B([i-1, i])$. While for ease of presentation we focus on this specific model, in fact, our results apply to general distributions satisfying certain continuity conditions.[6]

[4]This has also been observed, e.g., in [114].

[5]While this appears to be a more restrictive definition than the common notion of piecewise constant distributions, observe that without loss of generality, one can always scale the pieces and the distributions simultaneously, such that the pieces are of the same length.

[6]For example, it is known that all Lebesgue measurable functions (including all continuous ones) are approximated by step functions (i.e., piecewise constant ones) up to any precision. This gives a way of generalizing our results to all Lebesgue measurable density functions.

### 7.5.2 Accurate Classification with Continuous Outcomes

Under the continuity assumption, we now show that perfect accuracy is possible with fixed cost per test, whenever the good and bad distributions, $G$ and $B$, are not identical. Moreover, as in the variable cost case, perfect accuracy again can be achieved using a memoryless policy.

**Theorem 7.4.** *When the outcome space $O = [0, k]$, for any cost per test $0 < c < 1$, and good and bad distributions $G$ and $B$ (where $G \neq B$) that are constant on $[i-1, i]$ for any $i \in [k]$, there exists a policy $\mathcal{P}$ such that $(\mathcal{P}, c)$ is perfectly accurate for $G$ and $B$. Moreover, $\mathcal{P}$ consists of only singleton sets of outcomes.*

### 7.5.3 Nearly Optimal Policies

As illustrated by Theorem 7.2, memoryless policies do not generally achieve optimal efficiency when the cost per test is set by the principal. The same intuition applies to the fixed cost case as well. Below, we construct a policy for any piecewise constant and distinct good and bad distributions which requires at most $\lfloor 1/c \rfloor + 1$ tests, where $c$ is the cost per test. We then show that the policy we construct has (1) optimal worst case efficiency, and (2) approximately optimal expected efficiency when the good and bad distributions are not trivially different.

**Theorem 7.5.** *When the outcome space is $O = [0, k]$, for any cost per test $0 < c < 1$, and good and bad distributions $G$ and $B$ (where $G \neq B$) that are constant on $[i-1, i]$ for any $i \in [k]$, there exists a policy $\mathcal{P}$ such that $(\mathcal{P}, c)$ is perfectly accurate for $G$ and $B$. Moreover,*

$$\Pr[T(\mathcal{P}, G) \leq \lfloor 1/c \rfloor + 1] = 1.$$

Unlike Theorem 7.2, the above policy-cost pair is not guaranteed to dominate all other perfectly accurate pairs. However, it is in fact optimal in terms of the maximum number of tests a good agent may have to take before getting accepted, as long as the good and bad distributions share the same support.

**Proposition 7.1.** *For any piecewise constant good and bad distributions $G$ and $B$ where $G$ and $B$ share the same support, if a policy-cost pair $(\mathcal{P}, c)$ is perfectly accurate, then*

$$\Pr[T(\mathcal{P}, G) < \lfloor 1/c \rfloor + 1] < 1.$$

*In other words, the maximum number of tests a good agent may have to take is at least $\lfloor 1/c \rfloor + 1$.*

Proposition 7.1 states that the policy constructed in Theorem 7.5 is in fact optimal in terms of the maximum number of tests any good agent may have to take. However, it is unclear whether one can do significantly better[7] in terms of the expected number of tests, especially when the good and bad distributions are sufficiently different. We do show that, even when good and bad distributions are far apart (i.e., when $d_{\mathrm{TV}}(G, B) = \Omega(1)$), there still exist good and bad distributions $G$ and $B$ such that the expected number of tests required is at least $1/2c$. In other words, the policy

[7]It is certainly possible to do somewhat better. For example, when $G([0,1]) = B([1,2]) = 0.9$, $G([1,2]) = B([0,1]) = 0.1$, and the cost per test $c = 0.5$, $\mathcal{P} = \{\{o\} \mid o \in [0,1]\}$ accepts a good agent after $10/9 < \lfloor 1/c \rfloor + 1 = 3$ tests in expectation. $(\mathcal{P}, c)$ is perfectly accurate, because a bad agent will require 10 tests in expectation, so a bad agent will not attempt the test.

constructed in Theorem 7.5 is also asymptotically optimal (in a worst-case sense) in terms of the expected number of tests.

**Proposition 7.2.** *There exist piecewise constant good and bad distributions $G$ and $B$ where $d_{\mathrm{TV}}(G, B) \geq 0.1$, such that if a policy-cost pair $(\mathcal{P}, c)$ is perfectly accurate, then*

$$\mathbb{E}[T(\mathcal{P}, G)] \geq \frac{1}{2c}.$$

Again, one can contrast Theorem 7.5 with the case where the principal directly determines how many tests an agent takes. In that case, as discussed above, the number of tests required to correctly identify an agent's distribution $D$ is $\Omega(d_{\mathrm{TV}}(G, B)^{-2})$, whereas Theorem 7.5 requires good agents to take at most about $1/c$ tests and guarantees perfect accuracy, regardless of how close $G$ and $B$ are to each other. In other words, unless $G$ and $B$ are far away and a small error probability is acceptable, allowing agents to choose between taking tests and leaving immediately is far more efficient than enforcing a certain number of tests.

Finally, we remark that with a fixed cost per test, there is no tradeoff between efficiency (i.e., the number of tests taken by good agents) and cost efficiency (i.e., the expected total cost paid by good agents) — they are always proportional to each other.

## 7.6 Conclusion and Future Research

In this chapter, we characterize the accuracy and efficiency of classification with optional tests. Our results partially explain the practical success of optional tests, and provide a principled way of designing accurate and efficient classification processes. In particular, we show how much better one can do with self-selection than in comparable settings without self-selection that were studied recently, even when we augment those models with self-selection in the simplest possible way. Our results also easily generalize to some richer settings. For example, if taking the test might make one better at the test next time (due to practice), this retains the key property that once an agent starts taking tests, that agent will continue until the agent succeeds. For future directions, one could relax some of the assumptions to obtain more robustness in the design of classification processes. For example, test outcomes might be strategically transformed (as studied in [113], Chapter 6 of this dissertation), the cost per test might be unknown, and agents might not be completely sure about their own distributions before taking tests.

## 7.7 Omitted Proofs

*Proof of Lemma 7.1.* Suppose the agent's distribution is $D$. Let $\pi : \mathcal{S} \to \{\mathsf{T}, \mathsf{L}\}$ be any strategy of the agent (which specifies the action to take at each state), and $V : \mathcal{S} \to \mathbb{R}_+$ be the expected onward utility of the agent by playing $\pi$, i.e.,

$$V(S) = \begin{cases} \mathbb{I}[\mathcal{P} \text{ accepts } S], & \text{if } \pi(S) = \mathsf{L} \\ -c + \mathbb{E}_{o \sim D}[V(S \cup \{o\})], & \text{otherwise.} \end{cases}$$

$\mathbb{I}[\cdot]$ above denotes the indicator of a statement. Suppose $\pi$ does not play $\mathsf{L}$ immediately, or keeps playing $\mathsf{T}$ till the set of outcomes can be accepted. Formally, suppose $V(\emptyset) > 0$ (which is possible only when $\pi(\emptyset) = \mathsf{T}$), and there exists $S$ not accepted by $\mathcal{P}$, where $\pi(S) = \mathsf{L}$ (so $V(S) = 0$).[8] Without loss of generality, let $S$ be a state with the smallest cardinality among those satisfying this condition, which implies that for any $S' \subsetneq S$, $\pi(S') = \mathsf{T}$. We show that such a policy cannot be optimal, which implies the claim to be proved.

Consider an extended version of the original MDP, where the state space is all ordered sequences of outcomes. At state $(o_1, \dots, o_t)$ (here the notation $(\dots)$ emphasizes the fact that the sequence of outcomes is ordered), if action $\mathsf{L}$ is played, the MDP terminates and the agent gains reward $\mathbb{I}[\mathcal{P} \text{ accepts } \{o_1, \dots, o_t\}]$. If action $\mathsf{T}$ is played, the agent gains reward $-c$ and transitions to state $(o_1, \dots, o_t, o_{t+1})$, where $o_{t+1} \sim D$. Observe that the optimal expected reward in the extended MDP is the same as that in the original MDP. So, to show that $\pi$ is suboptimal in the original MDP, we only need to construct a strategy in the extended MDP with higher expected reward. Consider the following policy $\pi'$ in the extended MDP, which mimicks $\pi$ unless the first $|S|$ outcomes are exactly $S$, in which case it drops these outcomes and starts over.

$$\pi'((o_1, \dots, o_t)) = \begin{cases} \pi(\{o_1, \dots, o_t\} \setminus S), & \text{if } \{o_1, \dots, o_{|S|}\} = S \\ \pi(\{o_1, \dots, o_t\}), & \text{otherwise.} \end{cases}$$

Consider the expected reward $V'$ of $\pi'$. First observe that for any state $(o_1, \dots, o_t)$, if $t \geq |S|$ and $\{o_1, \dots, o_{|S|}\} \neq S$, then $V'((o_1, \dots, o_t)) = V(\{o_1, \dots, o_t\})$. We now argue $V'((o_1, \dots, o_t)) > 0 = V(\{o_1, \dots, o_t\})$ when $\{o_1, \dots, o_t\} = S$. We couple the onward outcomes from $(o_1, \dots, o_t)$ in the extended MDP with the sequence of outcomes in the original MDP (both are iid draws from $D$), and argue that for any such sequence $o_{t+1}, \dots, o_{t'}$, the reward collected by $\pi'$ on this sequence is at least the reward collected by $\pi$ starting from the initial state $\emptyset$. There are essentially two cases.

- $\pi$ keeps taking tests on $o_{t+1}, \dots, o_{t'}$. Here, both $\pi$ and $\pi'$ gain reward $-c \times (t' - t - 1)$.
- $\pi$ decides to leave after receiving outcome $o_{t''}$, where $t + 1 \leq t'' \leq t'$. Here, $\pi$ gains reward

  $$-c \times (t'' - t - 1) + \mathbb{I}[\mathcal{P} \text{ accepts } \{o_{t+1}, \dots, o_{t''}\}].$$

  On the other hand, $\pi'$ gains reward

  $$-c \times (t'' - t - 1) + \mathbb{I}[\mathcal{P} \text{ accepts } \{o_1, \dots, o_{t''}\}],$$

  which is at least the reward $\pi$ gains, since $\{o_1, \dots, o_{t''}\} \supseteq \{o_{t+1}, \dots, o_{t''}\}$.

So in any case $\pi'$ collects no less reward than $\pi$ on $o_{t+1}, \dots, o_{t'}$, and as a result,

$$V'((o_1, \dots, o_t)) \geq V(\emptyset) > 0.$$

Summarizing the above, for any state $(o_1, \dots, o_t)$ where $t = |S|$, we have $V(\{o_1, \dots, o_t\}) \leq V'((o_1, \dots, o_t))$, and the inequality is strict when $\{o_1, \dots, o_t\} = S$. Now we can apply backward

[8] This leaves the case where $V(\emptyset) = 0$ and $\pi(\emptyset) = \mathsf{T}$. However, in that case, playing $\mathsf{L}$ immediately is also an optimal strategy.

induction, and show that for any $(o_1, \ldots, o_t)$ where $t < |S|$, $V(\{o_1, \ldots, o_t\}) \leq V'((o_1, \ldots, o_t))$, and the inequality is strict when $\{o_1, \ldots, o_t\} \subseteq S$. This is possible since by the choice of $S$, for any such state, $\pi(\{o_1, \ldots, o_t\}) = \pi'((o_1, \ldots, o_t)) = \mathsf{T}$. In particular, we have $V(\emptyset) < V'(())$ (i.e., $V'$ of the empty sequence of outcomes), as desired. This implies that $\pi$ is suboptimal in the original MDP, and concludes the proof. □

*Proof of Lemma 7.2.* Consider the optimal strategy of an agent with distribution $D$. In light of Lemma 7.1, we only need to consider the expected reward when the agent keeps taking tests until he is accepted, and compare that against $0$, which is the reward of leaving immediately. If the agent keeps taking tests, his expected cumulative reward is precisely

$$\mathbb{E}_{\{o_t\}}[1 - c \cdot T(\mathcal{P}, D)].$$

So, taking tests until acceptance is more preferable iff this number is strictly greater than $0$, i.e.,

$$c \cdot \mathbb{E}_{\{o_t\}}[T(\mathcal{P}, D)] < 1,$$

which is precisely the condition in the claim. □

*Proof of Lemma 7.3.* When the condition holds, one may choose any

$$c \in (1/\mathbb{E}[T(\mathcal{P}, B)], 1/\mathbb{E}[T(\mathcal{P}, G)]) .$$

Lemma 7.2 then guarantees that $(\mathcal{P}, c)$ is perfectly accurate. When the condition does not hold, for any $c \geq 0$, we always have

$$c \cdot \mathbb{E}[T(\mathcal{P}, G)] \geq c \cdot \mathbb{E}[T(\mathcal{P}, B)],$$

and by Lemma 7.2 such a pair $(\mathcal{P}, c)$ cannot be perfectly accurate. □

*Proof of Theorem 7.1.* We prove the theorem by construction. The idea is to focus on some particular outcome $o^*$ where $G(o^*) > B(o^*)$ and let $\mathcal{P} = \{\{o^*\}\}$. So for each test, the probability that a good agent gets outcome $o^*$ and therefore gets accepted is strictly greater than the same probability for a bad agent. The principal can then set the cost per test $c$ properly so that good agents strictly prefer taking tests, while bad agents strictly prefer leaving directly.

First, since $G \neq B$ and $O = [k]$ is finite, there exists some $o^* \in O$ such that $G(o^*) > B(o^*) \geq 0$. Let $P = \{o^*\}$ as stated above. Consider the situation of good agents. Each time a good agent takes a test, the probability that he receives outcome $o^*$ is $G(o^*)$, so $T(\mathcal{P}, G)$ follows a geometric distribution with parameter $G(o^*)$, and we have

$$\mathbb{E}[T(\mathcal{P}, G)] = G(o^*)^{-1}.$$

Similarly, for bad agents we have

$$\mathbb{E}[T(\mathcal{P}, B)] = B(o^*)^{-1} > G(o^*)^{-1} = \mathbb{E}[T(\mathcal{P}, G)].$$

By Lemma 7.3, this immediately implies that $\mathcal{P}$ is perfectly implementable. □

*Proof of Theorem 7.2.* We first show the easy part, i.e., $\mathcal{P}$ is perfectly implementable. By Lemma 7.3, this is equivalent to

$$\mathbb{E}[T(\mathcal{P}, G)] < \mathbb{E}[T(\mathcal{P}, B)].$$

Below we compute both. Observe that for any distribution $D$ over $O$,

$$\Pr[T(\mathcal{P}, D) \leq 2] = 1.$$

For $G$,

$$\Pr[T(\mathcal{P}, G) = 1] = \Pr_{o \sim G}[o \in P] = G(P).$$

So

$$\mathbb{E}[T(\mathcal{P}, G)] = G(P) + 2(1 - G(P)) = 2 - G(P).$$

Similarly,

$$\mathbb{E}[T(\mathcal{P}, B)] = 2 - B(P),$$

which by the choice of $P$ is strictly larger than $2 - G(P)$. This guarantees that $\mathcal{P}$ is perfectly implementable.

Now consider any perfectly implementable policy $\mathcal{P}'$. We show that for any $t \in \mathbb{Z}_+$,

$$\Pr[T(\mathcal{P}, G) \leq t] \geq \mathbb{E}[T(\mathcal{P}', G) \leq t].$$

Since

$$\Pr[T(\mathcal{P}, G) \leq 2] = 1,$$

we only need to show that

$$\Pr[T(\mathcal{P}, G) = 1] \geq \mathbb{E}[T(\mathcal{P}', G) = 1].$$

Suppose towards a contradiction the opposite. We argue below that $\mathcal{P}'$ cannot be perfectly implementable.

Let

$$P' = \{o \mid \mathcal{P}' \text{ accepts } \{o\}\}.$$

By our assumption on $\mathcal{P}'$,

$$\begin{aligned} G(P) = \Pr_{o \sim G}[\mathcal{P} \text{ accepts } \{o\}] &= \Pr[T(\mathcal{P}, G) = 1] \\ &< \mathbb{E}[T(\mathcal{P}', G) = 1] = \Pr_{o \sim G}[\mathcal{P}' \text{ accepts } \{o\}] = G(P'). \end{aligned}$$

Then by the choice of $P$, we have $G(P') \leq B(P')$. We argue below that this implies

$$\mathbb{E}[T(\mathcal{P}', G)] \geq \mathbb{E}[T(\mathcal{P}', B)],$$

and as a result, $\mathcal{P}'$ is not perfectly implementable.

In fact, we show an even stronger claim, i.e., for any $t \in \mathbb{Z}_+$,

$$\Pr[T(\mathcal{P}', G) \leq t] \leq \Pr[T(\mathcal{P}', B) \leq t].$$

First observe that

$$G(P') = \Pr[T(\mathcal{P}', G) = 1] \leq \Pr[T(\mathcal{P}', B) = 1] = B(P').$$

To analyze $\Pr[T(\mathcal{P}', G) = t]$ and $\Pr[T(\mathcal{P}', B) = t]$ for $t > 1$, we need to take a closer look at $P'$, $G$ and $B$. Without loss of generality, assume $P' = [k'] = \{1, \ldots, k'\}$ for some $k' \leq k = |O|$. Moreover, if $k' = k$, then

$$\Pr[T(\mathcal{P}', G) = 1] = \Pr[T(\mathcal{P}', B) = 1] = 1,$$

and we are done. From now on we assume $k' < k$. Observe that for any $i \in \{k' + 1, \ldots, k\}$, $G(i) \geq B(i)$. This is because otherwise the set $S_{-i} = [k] \setminus \{i\}$ satisfies $G(S_{-i})) > B(S_{-i})$ and $G(S_{-i}) \geq G(P') > G(P)$, violating the choice of $P$.

Consider two sequences of iid random outcomes, $\{g_t\}_t$ and $\{b_t\}_t$, drawn from $G$ and $B$ respectively. We couple $\{g_t\}_t$ and $\{b_t\}_t$ such that, when defined as functions of the two outcome sequences respectively, we always have

$$T(\mathcal{P}', G) \geq T(\mathcal{P}', B).$$

Let $\{\theta_t\}_t$ be iid real numbers distributed uniformly at random over $[0, 1]$, and

$$\alpha = \sum_{i \in [k]} \min(G(i), B(i)).$$

Let $g_t$ be defined in the following way.

$$g_t = \begin{cases} \min\left\{ i \in [k] \,\middle|\, \sum_{j \in [i]} \min(G(j), B(j)) \geq \theta_t \right\}, & \text{if } \theta_t \leq \alpha \\ \min\left\{ i \in [k] \,\middle|\, \sum_{j \in [i]} \max(G(j) - B(j), 0) + \alpha \geq \theta_t \right\}, & \text{otherwise.} \end{cases}$$

And similarly,

$$b_t = \begin{cases} \min\left\{ i \in [k] \,\middle|\, \sum_{j \in [i]} \min(G(j), B(j)) \geq \theta_t \right\}, & \text{if } \theta_t \leq \alpha \\ \min\left\{ i \in [k] \,\middle|\, \sum_{j \in [i]} \max(B(j) - G(j), 0) + \alpha \geq \theta_t \right\}, & \text{otherwise.} \end{cases}$$

For any $t \in \mathbb{Z}_+$, $g_t$ and $b_t$ satisfy the following properties.

- If $\theta_t \leq \alpha$, then $g_t = b_t$.
- If $\theta_t > \alpha$, then $b_t \in [k'] = P'$. This is because for any $i > k'$, $G(i) \geq B(i)$.

This immediately implies the desired claim. In fact, let $T$ be the smallest integer such that $\theta_T > \alpha$, so $g_t = b_t$ for any $t < T$. There are two cases.

- $\mathcal{P}'$ accepts $\{g_1, \ldots, g_{T-1}\} = \{b_1, \ldots, b_{T-1}\}$ (and maybe also some prefix of this sequence of outcomes). In such cases, $T(\mathcal{P}', G) = T(\mathcal{P}', B)$.
- Otherwise, we have $T(\mathcal{P}', G) \geq T$. On the other hand, $b_T \in P'$, so $T(\mathcal{P}', B) = T \leq T(\mathcal{P}', G)$.

So in any case, we have

$$T(\mathcal{P}', B) \leq T(\mathcal{P}', G).$$

This concludes the proof. □

*Proof of Theorem 7.3.* First we construct $(\mathcal{P}, c)$. Let $\mathcal{P} = \{\{o^*\}\}$ where $o^* \in O$ is an outcome such that $B(o^*)/G(o^*) = \min_{o \in O} B(o)/G(o)$. Note that whenever $G \neq B$, $B(o^*)/G(o^*) < 1$. Below we show that setting $c = B(o^*)$ gives the desired cost efficiency. First observe that

$$c \cdot \mathbb{E}[T(\mathcal{P}, B)] = c \cdot B(o^*)^{-1} = B(o^*) \cdot B(o^*)^{-1} = 1.$$

So by Lemma 7.2, bad agents never take tests. For good agents,

$$c \cdot \mathbb{E}[T(\mathcal{P}, G)] = B(o^*) \cdot G(o^*)^{-1} < B(o^*) \cdot B(o^*)^{-1} = 1.$$

So good agents always take tests, and pay expected cost $B(o^*)/G(o^*) < 1$.

Now consider any memoryless policy $\mathcal{P}' = \{\{o\} \mid o \in P'\}$, where $P' \subseteq O$. We have

$$B(P')/G(P') \geq \min_{o \in O} B(o)/G(o).$$

Suppose $(\mathcal{P}', c')$ is perfectly accurate. By Lemma 7.2, in order to make bad agents unwilling to take tests, we have

$$c' \geq B(P').$$

Then for good agents, the expected total cost can be bounded in the following way.

$$c' \cdot \mathbb{E}[T(\mathcal{P}', G)] \geq B(P')/G(P') \geq \min_{o \in O} B(o)/G(o),$$

which is the second half of the theorem. □

*Proof of Theorem 7.4.* Without loss of generality, suppose for any $i \in [k-1]$, $G(i)/B(i) \geq G(i+1)/B(i+1)$. (If this assumption does not hold, we can reorder the outcome space by $G(i)/B(i)$ and renumber the outcomes in the new order.) Note that since $G \neq B$, we must have $G(1) > B(1)$ and $G(k) < B(k)$, and as a result, for any $\theta \in (0, k)$, $G([0, \theta]] > B([0, \theta])$. Consider the family of policies $\mathcal{P}(\theta)$ parametrized by a threshold $\theta \in O$, defined as follows.

$$\mathcal{P}(\theta) = \{\{o\} \mid o \in [0, \theta]\}.$$

Observe that for any $\theta \in (0, k)$,

$$\Pr_{o \sim G}[\mathcal{P}(\theta) \text{ accepts } \{o\}] = G([0, \theta]) > B([0, \theta]) = \Pr_{o \sim B}[\mathcal{P}(\theta) \text{ accepts } \{o\}].$$

Since $\mathcal{P}(\theta)$ is memoryless, this further implies that for any $\theta \in (0, k)$,

$$\mathbb{E}[T(\mathcal{P}(\theta), G)] = G([0, \theta])^{-1} < B([0, \theta])^{-1} = \mathbb{E}[T(\mathcal{P}(\theta), B)].$$

Our goal is to show that there exists some $\theta \in (0, k)$, such that $(\mathcal{P}(\theta), c)$ is perfectly accurate, which by Lemma 7.2 is equivalent to

$$c \cdot \mathbb{E}[T(\mathcal{P}(\theta), G)] < 1 \leq c \cdot \mathbb{E}[T(\mathcal{P}(\theta), B)].$$

Observe that

$$\mathbb{E}[T(\mathcal{P}(\theta), B)] = B([0, \theta])^{-1}$$

is continuous in $\theta$ on $O$. Since $B([0, 0])^{-1} = \infty$ and $B([0, k])^{-1} = 1$, there must be some $\theta^* \in (0, k)$, such that

$$B([0, \theta^*])^{-1} = c^{-1}.$$

Moreover, the same $\theta^*$ satisfies

$$G([0, \theta^*])^{-1} < B([0, \theta^*])^{-1} = c^{-1}.$$

As a result, $\mathcal{P}(\theta^*)$ satisfies the desired conditions. This concludes the proof. □

*Proof of Theorem 7.5.* The plan is to consider a family of policies parametrized by some threshold, each of which has the desired worst-case efficiency, and argue some policy in this family together with the cost per test $c$ achieves perfect accuracy.

Let $T = \lfloor 1/c \rfloor + 1$. Again, without loss of generality, suppose for any $i \in [k-1]$, $G(i)/B(i) \geq G(i+1)/B(i+1)$, and therefore for any $\theta \in (0, k)$, we have $G([0, \theta]] > B([0, \theta])$. For $\theta \in O$, let

$$\mathcal{P}(\theta) = \{\{o\} \mid o \in [0, \theta]\} \cup \{\{o_1, \ldots, o_T\} \mid \forall t \in [T],\, o_t \in O\}.$$

That is, $\mathcal{P}(\theta)$ accepts any set of outcomes which contains some outcome from $[0, \theta]$, or has cardinality at least $T$. Observe that $\mathbb{E}[T(\mathcal{P}(\theta), B)]$ is continuous in $\theta$. Moreover,

$$\mathbb{E}[T(\mathcal{P}(0), B)] = T > c^{-1} \qquad \text{and} \qquad \mathbb{E}[T(\mathcal{P}(k), B)] = 1 < c^{-1}.$$

So, there must be some $\theta^* \in (0, k)$ such that

$$\mathbb{E}[T(\mathcal{P}(\theta^*), B)] = c^{-1}.$$

And since $G([0, \theta^*]) > B([0, \theta^*])$, for any $t < T$,

$$\Pr[T(\mathcal{P}(\theta^*), G) \leq t] > \Pr[T(\mathcal{P}(\theta^*), B)].$$

This is because $T(\mathcal{P}(\theta^*), G)$ and $T(\mathcal{P}(\theta^*), B)$ follow geometric distributions on $\{1, \ldots, T-1\}$, with parameters $G([0, \theta^*])$ and $B([0, \theta^*])$ respectively. And as a result,

$$\mathbb{E}[T(\mathcal{P}(\theta^*), G)] < \mathbb{E}[T(\mathcal{P}(\theta^*), B)] = c^{-1}.$$

By Lemma 7.2, $(\mathcal{P}(\theta^*), c)$ is perfectly accurate. □

*Proof of Proposition 7.1.* Without loss of generality assume the support of $G$ is $O = [0, k]$. Suppose, towards a contradiction, that

$$\Pr[T(\mathcal{P}, G) < \lfloor 1/c \rfloor + 1] = 1.$$

We show $(\mathcal{P}, c)$ cannot be perfectly accurate.

Let $T = \lfloor 1/c \rfloor + 1$. Consider the first $T - 1$ outcomes a good agent receives, $(g_1, \ldots, g_{T-1}) \sim G^{T-1}$. Note that the distribution of $(g_1, \ldots, g_{T-1})$, $G^{T-1}$, is piecewise constant over $O^{T-1}$, and its support is $O^{T-1}$. By our assumption, $\mathcal{P}$ accepts $(g_1, \ldots, g_{T-1})$ with probability 1. As a result, $\mathcal{P}$ must accept everything in $O^{T-1}$, potentially taking away a zero-measure set $S$ (where the measure is $G^{T-1}$, i.e., $G^{T-1}(S) = 0$).

Now since $B$ is piecewise constant over $O$, $B^{T-1}$ is also piecewise constant over $O^{T-1}$. We then have $B^{T-1}(S) = 0$. As a result, $\mathcal{P}$ accepts $(b_1, \ldots, b_{T-1}) \sim B^{T-1}$ with probability 1. In other words, we have

$$\Pr[T(\mathcal{P}, B) < T] = 1.$$

We then proceed by two cases. When $1/c > \lfloor 1/c \rfloor$, we have

$$\mathbb{E}[T(\mathcal{P}, B)] \leq \lfloor 1/c \rfloor \cdot \Pr[T(\mathcal{P}, B) \leq \lfloor 1/c \rfloor] \leq \lfloor 1/c \rfloor \cdot \Pr[T(\mathcal{P}, B) < T] < 1/c.$$

When $1/c = \lfloor 1/c \rfloor = T - 1$, since

$$\mathbb{E}[T(\mathcal{P}, G)] < 1/c = T - 1,$$

there must exist a multiset $T \in \mathcal{P}$ where $|T| \leq T - 2$. Then for bad agents, since $G$ and $B$ share the same support, we have $B^{T-2}(T) > 0$, and therefore

$$\mathbb{E}[T(\mathcal{P}, B)] \leq (T - 2) \cdot B^{T-2}(T) + (T - 1) \cdot (1 - B^{T-2}(T)) < T - 1 = 1/c.$$

So in any case, we have

$$\mathbb{E}[T(\mathcal{P}, B)] < 1/c.$$

Lemma 7.2 then implies that bad agents have the incentive to take tests till acceptance, and $(\mathcal{P}, c)$ is not perfectly accurate, a contradiction. □

*Proof of Proposition 7.2.* Below we construct $G$ and $B$. Let $k = 2$, and $G$ and $B$ be such that $G(1) = 0.4$, $G(2) = 0.6$, and $B(1) = B(2) = 0.5$. Observe that $d_{\mathrm{TV}}(G, B) = 0.1$. We argue that for any policy $\mathcal{P}$, we always have

$$\mathbb{E}[T(\mathcal{P}, B)] \leq 2\mathbb{E}[T(\mathcal{P}, G)].$$

This directly implies the proposition, since if $(\mathcal{P}, c)$ is perfectly accurate, then by Lemma 7.2,

$$\mathbb{E}[T(\mathcal{P}, G)] \geq \frac{1}{2}\mathbb{E}[T(\mathcal{P}, B)] > \frac{1}{2} \cdot c^{-1} = \frac{1}{2c}.$$

In order to bound $\mathbb{E}[T(\mathcal{P}, B)]$, again we consider the sequences of outcomes a good agent and a bad agent get respectively, $\{g_t\}_t$ and $\{b_t\}_t$. We again couple $\{g_t\}$ and $\{b_t\}$ such that

$$\Pr[T(\mathcal{P}, B) \leq T(\mathcal{P}, G)] = 1,$$

which implies the desired bound. Let $\{\theta_t\}_t$ be iid uniform numbers in $[0, 1]$. For any $t \in \mathbb{Z}_+$, let $b_t = 2\theta_t$. The construction of $\{g_t\}_t$ is more involved. Each $g_t$ is determined collectively by $\theta_{2t-1}$ and $\theta_{2t}$. For any $t \in \mathbb{Z}_+$, let

$$g_t = \begin{cases} \theta_{2t}/0.8, & \text{if } \theta_{2t-1} \le 0.5 \text{ and } \theta_{2t} \le 0.8 \\ 1 + (\theta_{2t} - 0.8)/1.2, & \text{if } \theta_{2t-1} \le 0.5 \text{ and } \theta_{2t} > 0.8 \\ 1 + (\theta_{2t} + 0.2)/1.2, & \text{otherwise.} \end{cases}$$

One may check that $\{g_t\}$ are iid, and each $g_t$ has distribution $G$. More importantly, we always have

$$g_t \in \{b_{2t-1}, b_{2t}\}.$$

As a result, for any $t \in \mathbb{Z}_+$ we always have

$$\{g_1, \ldots, g_t\} \subseteq \{b_1, \ldots, b_{2t}\},$$

as multisets. Therefore, whenever $\mathcal{P}$ accepts $\{g_1, \ldots, g_t\}$, it also accepts $\{b_1, \ldots, b_{2t}\}$. This concludes the proof. □

# Chapter 8

# Efficient Algorithms for Planning with Participation Constraints

## 8.1 Introduction

In the previous two parts of the dissertation, we have investigated two important topics in strategic machine learning: strategic classification and distinguishing strategic agents with samples. In the last part of the dissertation (Chapters 8 and 9), we investigate a third important topic in this domain: dynamic decision-making with strategic agents.

*How do we keep users from leaving?* That is the question asked daily by service providers such as banks, phone carriers, cable networks, and internet streaming companies. Much of the depth of the question originates from its dynamic nature: services last over (normally an extensive period of) time, users can leave at almost any moment, and the cost and benefit of leaving vary depending on the situation. Consider cable networks: when a new user signs their first contract, the network typically offers a discounted rate for 6 or 12 months, and if the user switches to another network during that time, there will be an early termination fee. However, after the first several months, when the user has become attached to the network, the monthly rate increases to the normal amount. Similar strategies (free trials, sign-up bonuses, etc.) are used by almost all service providers, especially those conducting business over the internet, where it is easier for users to leave a provider and switch to another. While there are certainly many other considerations behind such strategies, arguably the main objective is to keep users around while generating as much revenue as possible.

Even in the simple example of cable networks, the dynamic nature of the problem already introduces some delicate tradeoffs: a low (or $0$) early termination fee would make it harder to keep the user in the first several months, but would also encourage the user to start the service in the first place (equivalently, prevent the user from leaving at the very beginning); similarly, a higher normal rate (which means higher revenue) may still be acceptable once the user becomes sufficiently attached, but anticipating this eventual higher normal rate at the outset, a new user may not sign the contract in the first place, or may leave before becoming attached to the network. In other words, the network's policy in a "state" not only affects whether the user would leave

in that state, but also affects the user's decision in all "previous" states. Moreover, although the network and the user have misaligned interests, they are by no means in a zero-sum situation: the network may spend extra effort on improving the quality of the service, which would cost the network, but benefit the user even more, so they have less incentive to leave — the question is, is that worthwhile?

The presence of these issues suggests that designing a business strategy should be viewed as a *planning* problem, where the network is the planner (or the *principal*), and the user is the *agent*. The principal decides what *action* to take in each possible situation (i.e., each *state*). The action gives the principal and the agent possibly different rewards, and brings the state to a possibly random new state, where another action will be taken. The agent does not have a voice in which actions to take in which states, but always has the option to *leave*, which is the rational move to make when the agent's expected onward utility is below $0$ (where without loss of generality, $0$ is the utility induced by the best outside option, taking into consideration the cost of leaving). The goal of the principal is to design a policy that maximizes the principal's utility *subject to participation constraints*, which require that the agent's expected onward utility in every possible state should be at least $0$.[1] Such participation constraints introduce a mechanism design flavor to the problem, which distinguishes it from the classical problem of planning in Markov Decision Processes (MDPs). In fact, the latter can be viewed as a special case of the former, where conceptually, the agent does not have the ability to leave. We can bring the classical case into the formalism here by making sure the agent's reward is always nonnegative, so the agent would never want to leave.

In this chapter, we study the problem of planning with participation constraints from a computational point of view. Our goal is to answer the following question:

> *Given all parameters of a dynamic environment (i.e., reward functions and transition probabilities), can we efficiently compute a policy that maximizes the principal's utility subject to participation constraints?*

### 8.1.1 Equivalent Variants

For further motivation, we now present some variants that result in the same technical problem, so that our techniques apply to them as well. The reader who is satisfied to keep the above motivation in mind can safely skip this subsection, as the remainder of the chapter is written in line with the above motivation.

Equivalently, we can also consider problems where the goal is not to prevent the agent from *leaving*, but rather the goal is to prevent the agent from *entering*. Such examples are reminiscent of problems in the security games literature [63, 96]. For example, suppose we wish to discourage young people from joining a gang. We consider a representative agent (young person) and assume

[1] Of course, sometimes it is more desirable for the principal to simply let the agent leave. Still, technically, the assumption that the principal never allows the agent's expected onward utility to be negative is without loss of generality. This is because we can extend the MDP with an "end" action from each state (where it is possible for the agent to have negative onward utility), which deterministically leads to an additional, absorbing state corresponding to the process having ended. From this state, no additional rewards will be obtained by either party. With these extensions, a policy that would result in the agent actually leaving the MDP at some point is equivalent to the policy that is the same except for, at that point, taking the "end" action within the MDP, ensuring zero onward utility.

that once he joins the gang, he can no longer leave it. We can plan various, generally costly, enforcement measures that reduce the expected onward utility of being in the gang. (Note that we cannot *condition* these actions on whether the agent has joined the gang, as we are generally unable to observe gang membership; we have to commit to taking the actions even if we believe the agent was successfully deterred from joining the gang.) In this case, the goal is to ensure that this expected onward utility always stays *nonpositive*, so the agent will not join the gang. Simply negating the agent's rewards thus brings us back to the problem considered before.

In this example, our actions do not affect the agent's utility *before* the agent enters (joins the gang), whereas in the original problem we introduced, our actions do not affect the agent's utility *after* the agent leaves. A natural generalization is that our actions may affect the agent's utility both before and after the agent has left or entered. In the previous example, the most effective way to prevent the agent from joining the gang may not be to reduce the utility of being in the gang through enforcement, but rather to increase the utility of *not* being in the gang, for example by investing in after-school programs. Or, perhaps a combination of both is optimal. In this case, there is no longer a sharp distinction between entering and leaving — entering the gang is equivalent to leaving the alternative activities. What matters is the *difference* in reward between having entered/left and not yet having done so. If we normalize the rewards to the agent so that leaving results in onward rewards of $0$, we arrive back at the problem considered before.[2] Thus, in the remainder of this chapter, we focus on the original problem where leaving results in onward utility zero, in the understanding that this problem captures the full generality of such problems where rewards may be received both before and after leaving/entering.

### 8.1.2 Our Results

Our main result is an affirmative answer to our main question: *there is a polynomial-time algorithm that computes an optimal policy subject to participation constraints* (see Theorem 8.1). Simple as it may appear, we find the existence of such an algorithm highly counterintuitive. In classical MDPs, it is well known that optimal policies are without loss of generality deterministic and history-independent. Given this, an optimal policy can be found by a simple backward induction procedure. Unfortunately, this is no longer true in the presence of participation constraints. In fact, as we show in Section 8.2.2, restricting the policy to be either deterministic or history-independent may lead to an enormous loss in the principal's utility. In other words, to solve our problem, we need to optimize over randomized and history-dependent policies. Such optimization problems are often extremely hard (i.e., APX-hard or PSPACE-hard), which is the case for, e.g., partially observable MDPs and various special cases thereof [75, 81]. Another concrete example is that computing an optimal dynamic mechanism (a problem closely related to ours, which can be viewed as our setting with additional incentive-compatibility constraints) is APX-hard [115]. In fact, to the best of our knowledge, no other planning problem of a similar level of generality (i.e., generalizing planning in classical MDPs) where history-dependence is required admits efficient

[2]Indeed, previously, when we assumed staying out of the gang gave rewards of $0$, and we then negated the rewards of being in the gang, this corresponded exactly to this normalization step: the negated rewards of being in the gang are the normalized rewards of staying out of the gang in that case.

exact algorithms — this phenomenon is famously known as the *curse of history* [85, 95, 102]. Moreover, the fact that optimal policies may be history-dependent also rules out the possibility of computing the flat representation of an optimal policy efficiently, since the size of such a representation is already exponential in the number of states. (Our algorithm computes a succinct and implicit representation that encodes an optimal policy.) Given all the above, at least we were surprised that an efficient algorithm exists for planning with participation constraints.

Technically, our algorithm operates over the concept of Pareto frontier curves (formally defined in Section 8.2.3). Roughly speaking, the Pareto frontier curve associated with a state specifies, for each given onward utility that we may wish to guarantee the agent, the maximum onward utility for the principal that is achievable by a policy that satisfies: (1) it gives the agent exactly the desired onward utility and (2) it satisfies all future participation constraints. If we were able to somehow compute the Pareto frontier curves in all states, then it would be possible (although probably still nontrivial) to construct an optimal policy given these curves, or at least find the principal's optimal utility subject to participation constraints. However, although these curves are piecewise linear, in general they have exponentially many pieces, which makes computing them explicitly impossible. Our algorithm instead only tries to *evaluate* these curves in specific ways. In particular, we make two types of evaluations: evaluations at specific points, and evaluations along specific directions. While none of these evaluations can be done in a straightforward way (because we cannot compute the curves), we show that they can be recursively reduced to each other, through binary searching over the direction of an evaluation. Then, by scheduling all recursive evaluations in the right order, the algorithm is able to perform all essential evaluations using only polynomial computation, *given that the binary searches only require polynomially many iterations*. Bounding the number of iterations then requires a careful analysis of the numerical precision of the algorithm and the numerical "resolution" of the Pareto frontier curves, which turns out to work exactly in the way we want. As a result, we obtain a weakly polynomial-time algorithm (similar to all currently known polynomial-time algorithms for linear programming), whose time complexity depends on the number of bits required to encode the input numbers. A more detailed overview is given in Section 8.3.1.

The algorithm discussed above is for finite-horizon (episodic) environments, but it is not too hard to adapt it into an algorithm for infinite-horizon discounted environments. As a byproduct of our main result, we also give an algorithm that computes a policy that is additively suboptimal by at most $\varepsilon$ for any $\varepsilon > 0$ in infinite-horizon discounted environments, which runs in time polynomial in $\log(1/\varepsilon)$ and the size of the input. This is discussed in Section 8.3.4, together with other remarks and extensions of the finite-horizon algorithm.

## 8.2 Preliminaries

We first formally introduce the problem setup, discuss why the problem is challenging, and introduce the notion of Pareto frontier curves which will be instrumental in the algorithm and the analysis thereof.

### 8.2.1 Problem Setup

**The environment.** We mostly focus on finite-horizon environments in this chapter. There are $n$ states $\mathcal{S} = [n] = \{1, \dots, n\}$ and $m$ actions $\mathcal{A}$.[3] For each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, let $r_P(s, a)$ and $r_A(s, a)$ be the rewards of the principal and the agent respectively when action $a$ is played in state $s$. Moreover, let $P(s, a) \in \mathbb{R}^n$ be the transition probabilities when action $a$ is played in state $s$, where $P(s, a, s')$ is the probability that the next state is $s' \in \mathcal{S}$.

Without loss of generality, we assume that the states are ordered by reachability. Formally, for any $s, s' \in \mathcal{S}$ where $s \geq s'$, we have $P(s, a, s') = 0$ for all $a \in \mathcal{A}$.[4] We assume $s_{\text{init}} = 1$ is the initial state, and $s_{\text{term}} = n$ is the terminal state where no action is available.

**Histories and policies.** A history of length $t \in \mathbb{N}$ is a tuple $(s_1, a_1, \dots, s_t, a_t)$. Let $\mathcal{H}_t$ be the set of all histories of lengths $t$ for each $t \in \mathbb{N}$. In particular, $\mathcal{H}_0 = \{\emptyset\}$, where $\emptyset$ denotes the empty history. Let $\mathcal{H} = \bigcup_{t \in \mathbb{N}} \mathcal{H}_t$. For history $h = (s_1, a_1, \dots, s_t, a_t) \in \mathcal{H}$ and state-action pair $(s, a) \in \mathcal{S} \times \mathcal{A}$, we write $h + (s, a)$ for the history obtained by appending $(s, a)$ to the end of $h$:

$$h + (s, a) = (s_1, a_1, \dots, s_t, a_t, s, a).$$

Define $(s, a)+h$ similarly. For two history-state pairs $(h, s)$ and $(h', s')$ where $h = (s_1, a_1, \dots, s_t, a_t)$ and $h' = (s'_1, a'_1, \dots, s'_{t'}, a'_{t'})$, we say $(h', s')$ extends $(h, s)$, or $(h', s') \supseteq (h, s)$, if $t' > t$, and $(s_1, a_1, \dots, s_t, a_t, s)$ is a prefix of $(s'_1, a'_1, \dots, s'_{t'}, a'_{t'}, s')$.

Let $\Delta(\mathcal{A})$ denote the probability simplex over $\mathcal{A}$. A policy $\pi : \mathcal{H} \times \mathcal{S} \to \Delta(\mathcal{A})$ maps a history $h \in \mathcal{H}$ and a state $s \in \mathcal{S}$ to a random action $a \in \mathcal{A}$, where $\pi(h, s, a)$ is the probability that $\pi$ plays action $a$ at history-state pair $(h, s)$. Let $\Pi$ be the set of all (randomized, history-dependent) policies, which may or may not satisfy participation constraints (defined below).

**Utility and participation constraints.** Under a policy $\pi$, the expected onward utility $u_P^\pi(h, s)$ of the principal at history-state pair $(h, s)$ can be defined in the following recursive way.

$$u_P^\pi(h, s) = \begin{cases} 0 & \text{if } s = s_{\text{term}}, \\ \mathbb{E}_{a \sim \pi(h,s), s' \sim P(s,a)}[r_P(s, a) + u_P^\pi(h + (s, a), s')] & \text{otherwise.} \end{cases} \tag{8.1}$$

The onward utility of the agent $r_A^\pi(h, s)$ can be defined similarly, with $u_P$ and $r_P$ replaced by $u_A$ and $r_A$ respectively. We say a policy is feasible if it satisfies participation constraints in all states. Throughout the chapter, we assume that there exists a feasible policy (e.g., the policy that maximizes the agent's utility). Our goal is to find a feasible policy that maximizes the principal's overall utility. Formally, we want to compute a policy $\pi$ that maximizes $u_P^\pi(\emptyset, s_{\text{init}})$, subject to the participation constraints that $u_A^\pi(h, s) \geq 0$ for all $(h, s) \in \mathcal{H} \times \mathcal{S}$. [5]

[3] We assume all actions are available in every non-terminal state. This is without loss of generality because if an action $a$ is not available in a state $s$, we can set $a$'s rewards and transition probabilities to be the same as any available action in $s$.

[4] This is without loss of generality for finite-horizon (episodic) environments because one can make a copy of each state for each time step. Then, copies of states at earlier times can only transition into copies at later times.

[5] Note that some history-state pairs may not be reachable with positive probability under a policy. For consistency, we enforce participation constraints for such pairs as well. This is without loss of generality, since if $(h, s)$ is not

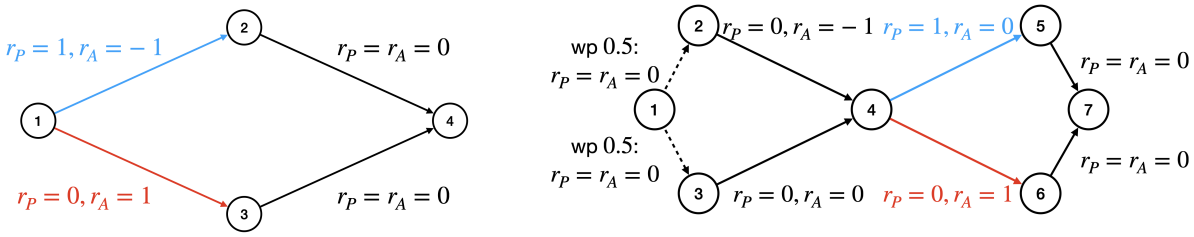


Figure 8.1: Examples where deterministic/history-independent policies are far from optimal.

**Encoding the input.** In order to properly formulate the computational problem, we assume that all parameters of the problem (including $n$, $m$, $r_P(s,a)$, $r_A(s,a)$, and $P(s,a,s')$) are given in binary representations. Moreover, we assume that $-1 \leq r_P(s,a) \leq 1$ and $-1 \leq r_A(s,a) \leq 1$ for all $s$ and $a$, and each of the input numbers has at most $L$ bits.

### 8.2.2 Some Natural Approaches and Why They Fail

Before diving into our algorithm, we first discuss some natural approaches and why they do not work. In classical MDPs, it is well known that optimal policies are without loss of generality deterministic and history-independent. Given this, an optimal policy can be found by a simple backward induction procedure. Unfortunately, this is no longer true in the presence of participation constraints. As we illustrate in the following examples, restricting the policy to be either deterministic or history-independent may lead to a significant loss in the principal's utility.

**Example 8.1.** Consider the left environment in Figure 8.1. This environment has $n = 4$ states, where $s_{\text{init}} = 1$ and $s_{\text{term}} = 4$. All states have at most 1 available action except for state 1. In state 1 there are two actions available, the upper (blue) one and the lower (red) one, leading to state 2 and state 3 respectively. The optimal (randomized) policy is to play the upper action and the lower action each with probability $1/2$ in state 1, which gives the principal overall utility $1/2$, and the agent onward utility 0 in all states. However, restricted to deterministic policies, the only feasible policy is to play the lower action in state 1, which gives the principal overall utility 0.

**Example 8.2.** Consider the right environment in Figure 8.1. This environment has $n = 7$ states, where $s_{\text{init}} = 1$ and $s_{\text{term}} = 7$. All states have at most 1 available action except for state 4. In state 4, there are two actions available, the upper (blue) one and the lower (red) one, leading to states 5 and 6 respectively. Moreover, in state 1, the only available action randomly transits to state 2 or 3 with equal probability. The optimal (history-dependent) policy is to play the upper action in state 4 if the previous state is state 3, and play the lower action if the previous state is state 2, which gives the principal overall utility $1/2$, and the agent nonnegative onward utility in all states. However, restricted to history-independent policies, the only feasible policy is to play the lower action in state 4, which gives the principal overall utility 0. In particular, note that in state 4 we cannot play one of the two actions uniformly at random, because then the agent's onward utility in

reachable, then the policy from this point onward does not affect the principal's utility, so we can run the policy that maximizes the agent's utility to satisfy participation constraints.

state 2 would be $-1/2$.

Optimizing over history-dependent policies is often computationally intractable. This phenomenon is famously known as the *curse of history* [85, 95, 102]. For instance, the problem of finding optimal policies for partially observable MDPs (as well as various special cases thereof [75, 81]) is PSPACE-hard. Another concrete example is that computing an optimal dynamic mechanism (which can be viewed as our setting with additional incentive-compatibility constraints) is APX-hard [115]. Another difficulty that arises from history-dependence is that we cannot efficiently describe an optimal policy in the flat representation, since the optimal policy may need to specify which action to take in each of exponentially many histories.

We conclude this section by showing that it is computationally hard to find an optimal deterministic policy. While the best deterministic policy could perform worse than the optimal randomized policy, there are situations where one may want to focus on deterministic policies. More importantly, this further illustrates the complexity of our problem. We reduce from the 0-1 knapsack problem.

**Claim 8.1.** *It is* NP*-hard to find an optimal deterministic policy that satisfies participation constraints.*

*Proof.* Consider a knapsack instance with $k$ items and size limit $S$, where item $i$ has size $s_i$ and value $v_i$. The goal of the knapsack problem is to pick a subset of items with maximum total value, subject to the constraint that their total size does not exceed $S$. Without loss of generality, assume $s_i \leq S$ for each $i \in [k]$. We construct an environment with $n = k + 2$ states that encodes the knapsack instance, where $s_{\text{init}} = 1$, $s_{\text{term}} = n$, and state $i + 1$ corresponds to item $i$.

There is a single action $a_0$ available in state $s_{\text{init}} = 1$ with $r_P(s_{\text{init}}, a_0) = 0$, $r_A(s_{\text{init}}, a_0) = \frac{1-k}{k} \cdot S$, and $P(s_{\text{init}}, a_0, i + 1) = \frac{1}{k}$ for each $i \in [k]$. For each item $i$, there are two actions $a_{i,0}$, $a_{i,1}$ available in the corresponding state $i + 1$. Intuitively, $a_{i,0}$ corresponds to not taking item $i$, where $r_P(i + 1, a_{i,0}) = 0$, $r_A(i + 1, a_{i,0}) = S$; and $a_{i,1}$ corresponds to taking item $i$, where $r_P(i+1, a_{i,1}) = v_i$, $r_A(i+1, a_{i,1}) = S-s_i$. Both actions lead to the terminal state deterministically, i.e., $P(i + 1, a_{i,0}, s_{\text{term}}) = P(i + 1, a_{i,1}, s_{\text{term}}) = 1$.

We show that this encodes the knapsack instance. Due to the structure of the environment we construct, all policies are without loss of generality history-independent, so we omit the dependence on $h$. For a deterministic policy $\pi$, let $T^\pi \subseteq [k]$ be the set of items that $\pi$ decides to pick, that is, $i \in T^\pi$ iff $\pi(s, a_{i,1}) = 1$. Then we have

- $r_P^\pi(s_{\text{init}}) = \dfrac{1}{k} \displaystyle\sum_{i \in T^\pi} v_i$.
- For each $i \in [k]$, we always have $r_A^\pi(i + 1) \geq 0$.
- $r_A^\pi(s_{\text{init}}) \geq 0 \iff \displaystyle\sum_{i \in T^\pi} s_i \leq S$.

It follows immediately that an optimal deterministic policy subject to participation constraints corresponds to an optimal solution to the knapsack instance. □

### 8.2.3 Pareto Frontier Curves

Now we define the notion of Pareto frontier curves, which is instrumental in designing and analyzing our algorithm. Intuitively, these curves capture the Pareto optimal tradeoffs between the principal's and the agent's (onward) utilities at different states.

We associate a Pareto frontier curve with each state $s \in \mathcal{S}$. For state $s$, we consider all policies starting at $s$ (as if $s$ is the initial state) and the onward utilities of the principal and the agent $u_A^\pi$ and $u_P^\pi$ as defined in Equation (8.1). We say a policy $\pi$ is *feasible in the future* iff $\pi$ satisfies the participation constraints at all later history-state pairs.

Let $D_s = [u_A^-(s), u_A^+(s)]$ be the range of onward utility of the agent that is achievable by policies that are feasible in the future. Formally,

$$u_A^-(s) = \min\{u_A^\pi(\emptyset, s) \mid \pi \in \Pi : u_A^\pi(h', s') \geq 0,\ \forall (h', s') \supseteq (\emptyset, s)\},$$
$$u_A^+(s) = \max\{u_A^\pi(\emptyset, s) \mid \pi \in \Pi : u_A^\pi(h', s') \geq 0,\ \forall (h', s') \supseteq (\emptyset, s)\}.$$

Note that we consider policies that satisfy participation constraints *after leaving* state $s$, and put no restrictions on the agent's onward utility *in* state $s$.

The Pareto frontier curve $f_s : D_s \to \mathbb{R}$ in state $s \in \mathcal{S}$ maps the agent's onward utility $x \in D_s$ to the maximum principal's onward utility $y$ that is achievable by some feasible-in-the-future policy $\pi$, such that the agent's onward utility is exactly $x$ under $\pi$. Formally, for each $s \in \mathcal{S}$ and $x \in D_s$,

$$f_s(x) = \max\{u_P^\pi(\emptyset, s) \mid \pi \in \Pi : u_A^\pi(\emptyset, s) = x \text{ and } u_A^\pi(h', s') \geq 0,\ \forall (h', s') \supseteq (\emptyset, s)\}. \quad (8.2)$$

The following property of Pareto frontier curves, which was observed in [118], plays an important role in our algorithm and analysis.

**Lemma 8.1.** *For each $s \in \mathcal{S}$, the Pareto frontier curve $f_s$ defined in Equation* (8.2) *is concave on $D_s$.*

The concavity of these curves is a direct consequence of the fact that randomizing between feasible-in-the-future policies always results in a feasible-in-the-future policy.

## 8.3 Our Algorithm and Analysis

Our main result is a polynomial-time exact algorithm for the problem of planning with participation constraints.

**Theorem 8.1.** *There is an algorithm that runs in time $\mathrm{poly}(n, m, L)$ and computes an optimal policy satisfying participation constraints, where $n$, $m$, and $L$ are the number of states, number of actions, and number bits required to encode each input number (rewards and transition probabilities) respectively.*

The proof of the theorem is deferred to Section 8.3.5, and the next subsection is dedicated to a more friendly presentation of the algorithm and the analysis.

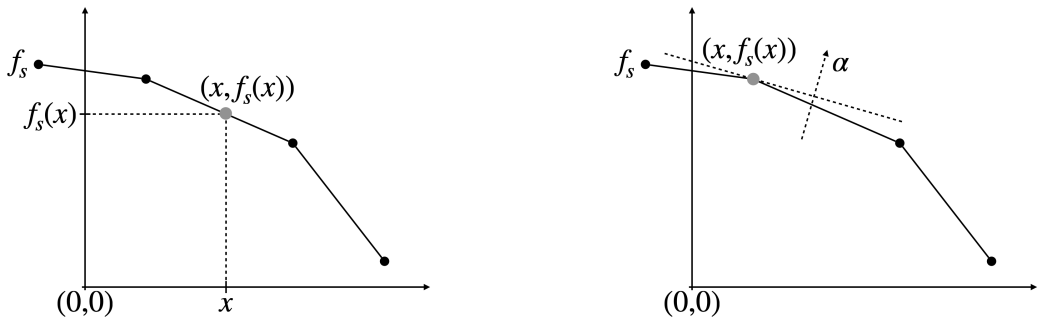


Figure 8.2: The two types of evaluation subroutines of our algorithm.

### 8.3.1 Overview of the Algorithm

Given the definition of Pareto frontier curves, the maximum overall utility of the principal that can be achieved by a feasible policy is equal to $\max_{x \in D_{s_{\text{init}}} \cap \mathbb{R}_+} f_{s_{\text{init}}}(x)$. [6] So, the problem of planning with participation constraints immediately reduces to computing the Pareto frontier curve at the initial state $s_{\text{init}}$ — which, unfortunately, turns out to be a highly challenging (if not impossible) task. In particular, although each $f_s$ is piecewise linear, there may be exponentially many pieces in each curve, which makes explicitly computing the curves infeasible. In [118], the authors circumvent this issue by allowing approximation — they give an approximation algorithm (which achieves an additive $\varepsilon$-approximation in $\text{poly}(1/\varepsilon)$ time) for planning with participation constraints by recursively computing approximations of the Pareto frontier curves, from later states to earlier ones. Their main technical contribution is identifying a computationally feasible recursive relation between the curves, and coming up with a way to approximate the curves using only a small number of pieces. However, it seems unlikely that similar approaches could lead to an efficient *exact* algorithm.

In contrast to their approach, our algorithm does not try to compute (or approximate) the entire Pareto frontier curves. Instead, we only evaluate the curves "at specific points" and "along specific directions" (see Figure 8.2). The left side of Figure 8.2 illustrates evaluating $f_s$ at a given point, where we want to compute $f_s(x)$ for a given $x$. The right side of Figure 8.2 shows an evaluation along a specific direction $\alpha \in \mathbb{R}^2$, which returns a point $(x, f_s(x))$ that maximizes the inner product $\alpha \cdot (x, f_s(x))$. These two types of evaluations correspond to the two major *conceptual* subroutines of our algorithm.

If these subroutines can be implemented efficiently, then we can immediately compute the maximum overall utility of the principal: it is equal to the $y$-coordinate of the point found by evaluating $f_{s_{\text{init}}}$ along the direction $(0, 1)$ if the $x$-coordinate of the returned point is nonnegative; otherwise it is equal to $f_{s_{\text{init}}}(0)$. This is true because $f_{s_{\text{init}}}$ is concave: if a point with the largest $y$-coordinate on $f_{s_{\text{init}}}$ is to the left of $x = 0$, then the optimal feasible point must have $x$-coordinate $0$.

Based on these observations, we only need to efficiently implement these two subroutines.

[6] We use $\mathbb{R}^+$ to denote the set of nonnegative real numbers.

Below we discuss how this is possible. We will refrain from being fully formal, and focus on the intuition instead. For the full description of the algorithm, see Algorithm 8.1.

**Evaluations at specific points.** Suppose we want to evaluate $f_s$ at $x$. We show that this can be reduced to multiple evaluations along specific directions of $f_s$. Consider the left side of Figure 8.3. To find the (gray) point $(x, f_s(x))$, we only need to find the two endpoints of the piece containing it, namely the (blue) point $(x_1, f_s(x_1))$ and the (green) point $(x_2, f_s(x_2))$. Then, taking the convex combination of these two endpoints with the right coefficients gives us $(x, f_s(x))$. These coefficients can be computed using $x$ (given as input), $x_1$ and $x_2$, as illustrated in Figure 8.3. So it suffices to find the two endpoints.

Consider, for example, the left endpoint $(x_1, f_s(x_1))$. There exists some direction $\alpha$ (e.g., $\alpha_1$ in the figure) such that

$$\alpha \cdot (x_1, f_s(x_1)) = \max_{x \in D_s} \alpha \cdot (x, f_s(x)).$$

We only need to find such an $\alpha$ and evaluate $f_s$ along that direction. To this end, observe that the maximizer found by evaluating along $\alpha$ moves on the curve monotonically as we rotate $\alpha$ (consider, from the left to the right, $\alpha_3, \alpha_1, \alpha_2$ and the corresponding maximizers, which are the red, blue, and green points respectively). Again this is because the curve is concave. So, we need to find the "rightmost" $\alpha$ such that the maximizer found by evaluating along $\alpha$ is to the left of $x$, i.e., the $x$-coordinate of that maximizer is no larger than $x$.

To achieve this, we perform a binary search over $\alpha$. We defer the discussion on the numerical issues of this binary search to Section 8.3.2. For now, we assume the number of iterations this binary search requires is $\mathrm{poly}(n, m, L)$, which is in fact the case, as we will show later.

**Evaluations along specific directions.** Now consider the other subroutine where we want to evaluate $f_s$ along a given direction $\alpha$. We show that this can be reduced to multiple evaluations of both types, of the Pareto frontier curves in later states. At a high level, evaluating $f_s$ along $\alpha$ can be viewed as a planning problem, where the goal is to find a policy $\pi$ that maximizes $\alpha \cdot (u_P^\pi(\emptyset, s), u_A^\pi(\emptyset, s))$, subject to participation constraints *in the future* (and not in state $s$).

Since the policy is unconstrained in state $s$, without loss of generality, an optimal policy $\pi$ has the Markovian property *in state $s$ only*: consider the behavior of the policy right after taking action $a$ in $s$, leaving $s$, and entering a later state $s' > s$. The subpolicy from this point on must maximize $\alpha \cdot (u_P^\pi((s,a), s'), u_A^\pi((s,a), s'))$ subject to participation constraints (including in state $s'$). In particular, the subpolicy at $s'$ does not depend on the action $a$ taken in state $s$ or the subpolicy in other later states.[7] This subpolicy corresponds to a point on $f_{s'}$, which can be found by evaluating $f_{s'}$ twice: along direction $\alpha$ and at $x = 0$ respectively, and then picking the point with the larger $x$-coordinate (again because $f_{s'}$ is concave). In other words, the planning subproblem in each state $s' > s$ can be reduced to two evaluations of $f_{s'}$. After solving these subproblems for each $s' > s$, the policy in state $s$ should choose an action $a$ which maximizes

$$\alpha \cdot (r_P(s,a), r_A(s,a)) + \mathbb{E}_{s' \sim P(s,a)} \left[ \alpha \cdot (u_P^\pi((s,a), s'), u_A^\pi((s,a), s')) \right].$$

[7]Note that the subpolicy in state $s'$ is in effect only if $s'$ is the first state reached after leaving $s$. In the case where we reach some $s''$ immediately after leaving $s$ and then later reach $s'$, it is the subpolicy in $s''$ that should apply.

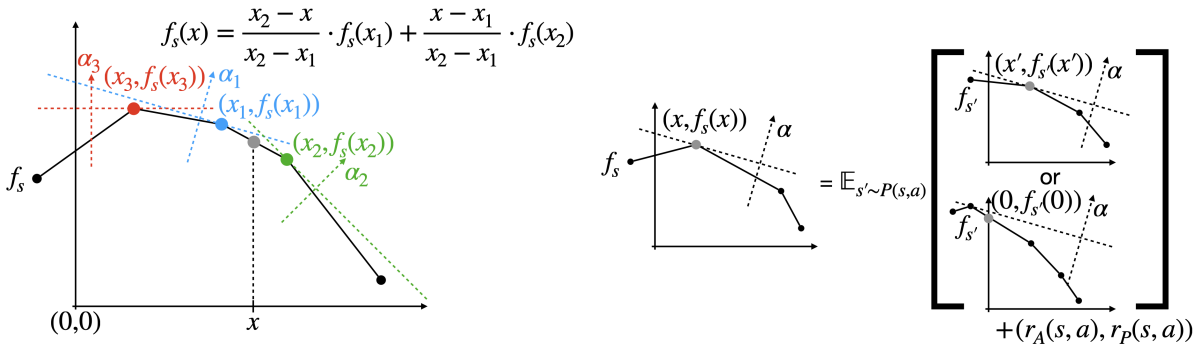


Figure 8.3: Recursive (naïve) implementations of the two subroutines.

The above procedure is illustrated in the right side of Figure 8.3, where the action $a$ is a maximizer of the above expectation. The two cases inside the expectation in the figure correspond to the two cases of the subproblem in each later state $s'$. The upper case is when evaluating $f_{s'}$ along $\alpha$ returns the (gray) point $(x', f_{s'}(x'))$ with $x' \geq 0$, so it corresponds to the subpolicy at $s'$. The lower case is when evaluating $f_{s'}$ along $\alpha$ gives a point with a negative $x$-coordinate, in which case the (gray) point $(0, f_{s'}(0))$ corresponds to the subpolicy in state $s'$.

**Putting everything together.** The above discussion already describes a way to perform both types of evaluations in *finite* time. This is because evaluations at specific points reduce to only evaluations along specific directions in the same state; and evaluations along specific directions reduce to only evaluations in later states (one evaluation of each type for each later state). However, a problem is that it generally takes exponential time if we recursively perform the evaluations for subproblems in the naïve way, because for both types of evaluations there can be polynomially many subproblems (i.e., the number of iterations in the binary search for evaluations at specific points, and the number of later states for evaluations along specific directions).

So, to evaluate $f_{s_{\text{init}}}$ at $x = 0$ and along $(0, 1)$ efficiently, we need to schedule and handle all the evaluations involved (most of which originate from recursive calls) in a more global manner. Intuitively, for any state $s \in \mathcal{S}$, we only ever need to calculate $f_s(0)$ and evaluate $f_s$ along a polynomial number of directions $\alpha$. This is because each direction $\alpha$ can be traced back to one of $n$ states that first queries for this $\alpha$; at the same time, each state only queries a polynomial number of different $\alpha$'s. Below we give an informal hierarchical description of the schedule, together with inline annotations.

- First observe that evaluations at $x = 0$ appear repeatedly in the naïve implementation. We therefore center our schedule around these evaluations.
- We will compute $f_s(0)$ for all states $s \in \mathcal{S}$ one by one from later states to earlier ones (i.e., from $s_{\text{term}} = n$ to $s_{\text{init}} = 1$), since the recursive dependence (as discussed above) never goes backwards. We call this the **outer loop**.
  - Consider some state $s$ in the outer loop, and suppose we have already computed $f_{s'}(0)$ for all $s' > s$. As discussed above, to compute $f_s(0)$, it suffices to perform a **binary search** on $\alpha$ in state $s$.

- In each iteration of the binary search, we need to perform an evaluation of $f_s$ along $\alpha$. As discussed above, we only need to evaluate $f_{s'}$ along $\alpha$ for each $s' > s$, since we already know $f_{s'}(0)$. This can be done in a single backward pass (from $s_{\text{term}} = n$ to $s + 1$) without nested recursive calls, which we call the **inner loop**.
  - For each $s' > s$ in the inner loop, the evaluation of $f_{s'}$ along $\alpha$ reduces to the evaluation of $f_{s''}$ along $\alpha$ and $f_{s''}(0)$ for all $s'' > s'$.
  - The former has already been computed in previous iterations of the inner loop, and the latter has already been computed in previous iterations of the outer loop. We only need to retrieve the two points for $s'' = s' + 1, \ldots, n$, which means every iteration of the inner loop takes $O(n)$ time.
- The inner loop has $O(n)$ iterations, so the total time is $O(n^2)$, which is also the runtime of one iteration of binary search.

▪ Now as discussed before, the binary search has $\text{poly}(n, m, L)$ iterations (we will come back to this not-yet-substantiated claim momentarily), so the total time is $\text{poly}(n, m, L)$.

- The outer loop has $O(n)$ iterations, so the total time of computing $f_s(0)$ for all $s \in \mathcal{S}$ is $\text{poly}(n, m, L)$.
- Finally, we need one last evaluation of $f_{s_{\text{init}}}$ along the direction $(0, 1)$. This can be done by a single call to the inner loop above, which takes time $O(n^2)$.

A formal description of our algorithm is given in Algorithm 8.1.

### 8.3.2 Handling Numerical Issues

Finally, we come back to the number of iterations required in the binary search in an evaluation at a specific point (used in the algorithm to compute $f_s(0)$ for each state $s$). We show that under appropriate parametrization of the direction $\alpha$, it suffices to perform the binary search up to some singly exponential precision, which implies the number of iterations is polynomial. In particular, we search over the slope of the perpendicular direction to $\alpha$. The intuition is that we only need to distinguish between the slopes of two consecutive pieces on $f_s$, which cannot be too close to each other. In fact, we will establish a stronger claim: the coordinates of all turning points on $f_s$ must be integral multiples of some singly-exponentially small quantity. Since these coordinates are bounded between $-n$ and $n$, the slope of the line between any two turning points (which do not even need to be adjacent) cannot take too many values. Moreover, the magnitude of the slope is upper bounded by some not too large quantity. This allows the binary search to terminate in not too many steps. We elaborate in the following paragraph.

To see why the above is true, we consider a specific procedure of recursively constructing the entire $f_s$ for all $s$ from later states to earlier ones, and treat all quantities involved in the construction as fractions. Fix some $s$, and assume $f_{s'}$ for any $s' > s$ has the desired property, i.e., the denominator of any quantity used to represent $f_{s'}$ is not too large. We argue that $f_s$ also has this property (where the denominator may be *moderately* larger than the denominators in later states — in fact, it is the blowup that we try to bound).

Recall that any turning point on $f_s$ can be found by evaluating along some direction. So fix a

turning point $(x, y)$ on $f_s$, and consider any direction $\alpha$ which gives this point. As discussed earlier (see the right side of Figure 8.3), there exists some action $a \in \mathcal{A}$ such that

$$\begin{aligned}(x, y) &= \mathbb{E}_{s' \sim P(s,a)}[(x_{s'}, y_{s'})] + (r_A(s, a), r_P(s, a)) \\ &= \sum_{s' > s} P(s, a, s') \cdot (x_{s'}, y_{s'}) + (r_A(s, a), r_P(s, a)),\end{aligned}$$

where $(x_{s'}, y_{s'})$ is either a turning point on $f_{s'}$ or $(0, f_{s'}(0))$. Among all quantities on the right hand side of the above equation, $P(s, a, s')$, $r_A(s, a)$, and $r_P(s, a)$ have at most $L$ bits in the binary representation, so the denominators of these quantities are at most $2^L$. Moreover, when $(x_{s'}, y_{s'})$ is a turning point on $f_{s'}$, by the induction hypothesis, the denominators of both coordinates are not too large. So if all $(x_{s'}, y_{s'})$ are turning points, then we immediately know that $(x, y)$ has the desired property: the denominator of both coordinates can only blow up by a factor of $2^L$. And importantly, in any case, the denominator of $x_s$, or $x_{s'}$ for any $s' > s$, can be at most $2^{nL}$, because the $x$-coordinates of turning points in any state only depend on the $x$-coordinates of turning points in later states.

The problematic case is when $(x_{s'}, y_{s'}) = (0, f_{s'}(0))$. To handle this case, we need to argue that the denominator of $y_{s'}$ is not too large either, compared to those of the turning points on $f_{s'}$. Recall that there exist turning points $(x_\ell, y_\ell)$ and $(x_r, y_r)$ on $f_{s'}$, where $x_\ell < 0$ and $x_r > 0$, such that

$$y_{s'} = \frac{x_r \cdot y_\ell - x_\ell \cdot y_r}{x_r - x_\ell}.$$

So the denominator of $y_{s'}$ (compared to that of $y_\ell$ or $y_r$) can blow up by at most a factor of $2^{nL}$ (which is the maximum denominator of $x_\ell$ and $x_r$), times the numerator of $x_r - x_\ell$ (which is upper bounded by $2n \cdot 2^{nL}$ because $-n \leq x_\ell < x_r \leq n$). So in any case, assuming $n \geq 2$, the maximum blowup incurred in the construction of $f_s$ is $2^{3nL}$, and consequently, the denominator of the $y$-coordinate of any turning point on the curve of any state is $2^{3n^2 L}$. The above discussion on numerical issues is formalized as Lemma 8.3, which is stated and proved in Section 8.3.5.

### 8.3.3 Decoding the Policy

Algorithm 8.1 outputs only an implicit representation of an optimal policy. In this subsection, we describe how to efficiently decode the output of Algorithm 8.1, so that we can execute the corresponding policy. The idea is to keep track of the current "objective direction", which is initially $(0, 1)$ and may randomly change (and in particular, rotate to the right) as the state evolves. This objective direction essentially corresponds to how much we need to compensate the agent from this point on in order to satisfy participation constraints. In other words, the objective direction succinctly encodes the relevant part of the history.

According to Algorithm 8.1, at any time, the onward policy in the current state (given the history) corresponds to either the maximizer along the objective direction, or the intersection of the Pareto frontier curve with $x = 0$. In the former case, the agent is satisfied with the current level of compensation, so we do not need to compensate more. In this case, we can take an action deterministically, and the objective direction does not change. In the latter case, we need to compensate

the agent more to satisfy participation constraints, so we rotate the objective direction to the right. In this case, we need to randomize between the two actions corresponding to the two endpoints of the piece containing the intersection point. Depending on which action is actually chosen, the new objective direction is the one for which the corresponding endpoint is the maximizer. Since all these points and directions (along with many other auxiliary points and directions) have been computed in Algorithm 8.1, we only need to read them from the output. The full algorithm is given as Algorithm 8.2, which maps each history-state pair to a (random) action. Algorithm 8.2 can be easily adapted into a dynamic procedure that plays the optimal policy on the fly while interacting with the environment (by updating $\alpha$), rather than re-analyzing the history from scratch at each point in time.

Note that the behavior of the policy output by Algorithm 8.1 is unspecified for some history-state pairs. However, if one strictly follows the specified part of the policy, then the unspecified part can never be reached (i.e., the probability that we arrive at such a history-state pair is $0$). For such unreachable pairs, the behavior of the decoding algorithm can be arbitrary.

### 8.3.4 Remarks and Extensions

**Structure of optimal policies.** Our algorithm also directly implies some structural properties of optimal policies with participation constraints. In particular:

- Although there might be exponentially many turning points on the Pareto frontier curves, for the optimal policy we compute, there are only polynomially many of them for which it is possible that the policy visits them. These points are maximizers for the polynomially many directions we queried during the computation of an optimal policy.
- Optimal policies are almost deterministic. In fact, an optimal policy randomizes between precisely $2$ actions when the participation constraint in the current state (given the history) is binding. This is also where the policy branches and history-dependence is introduced. In all other situations, the policy deterministically chooses an action. This aligns well with intuition: when no participation constraints are binding, it suffices to simply maximize the principal's utility, which naturally leads to a completely deterministic and history-independent policy.

**Extensions to richer constraints.** Our algorithm can be generalized to the case where the agent's onward utility in each state must be in one of several disjoint intervals (instead of a single interval $[0, \infty)$). Moreover, these feasible intervals can be different for each state. In order to handle multiple feasible intervals in a state $s$, we evaluate $f_s$ at the endpoints of all these intervals, which can be done by binary search. Once we have computed these points, to evaluate a curve along a direction $\alpha$, we only need to handle subproblems where we evaluate later curves *restricted to feasible intervals*. This can be done since if the unconstrained maximizer is infeasible, then the optimal feasible point must be one of the two endpoints that are closest to the unconstrained maximizer. Since we have already computed all endpoints, we can simply try the two points and choose the better one.

**Infinite-horizon environments with discounted reward.** Now we discuss how to extend our algorithm to the infinite-horizon case with discounted reward. For such environments, we describe an algorithm that computes a policy subject to participation constraints that is optimal up to an additive error of $\varepsilon > 0$, in time $\mathrm{poly}(n, m, L, \log(1/\varepsilon))$ for any $\varepsilon > 0$. This is done by reducing to the finite-horizon case and running Algorithm 8.1.

First we briefly define infinite-horizon environments. As in the finite-state case, there are $n$ states $\mathcal{S} = [n]$ and $m$ actions $\mathcal{A}$, and $r_P$, $r_A$ and $P$ denote the principal's reward, the agent's reward, and the transition probabilities respectively. There is an initial state $s_{\text{init}} = 1$, but no terminal state. We also do not require transitions to be from earlier states to later ones. In addition, there are discount factors $\delta_P \in (0, 1)$ and $\delta_A \in (0, 1)$ (which we treat as constants) for the principal and the agent respectively. Define histories similarly as in finite-horizon environments. The onward utility $u_P^\pi(h, s)$ of the principal under policy $\pi$ in state $s$ given history $h$ is defined recursively such that

$$u_P^\pi(h, s) = \mathbb{E}_{a\sim\pi(h,s),s'\sim P(s,a)}[r_P(s, a) + \delta_P \cdot u_P^\pi(h + (s, a), s')].$$

The onward utility $u_A^\pi$ of the agent is defined similarly, with $u_P^\pi$ and $r_P$ replaced with $u_A^\pi$ and $r_A$. Participation constraints require that for all $(h, s) \in \mathcal{H} \times \mathcal{S}$, $u_A^\pi(h, s) \geq 0$. We say a policy $\pi$ is feasible if it satisfies participation constraints. The goal is to find a feasible policy that maximizes the principal's overall utility $u_P^\pi(\emptyset, s_{\text{init}})$.

Note that the finite-horizon case can be viewed as a special case of the infinite-horizon case with discounted reward, by scaling the rewards appropriately and replacing the terminal state with an absorbing state from which there are no more rewards. As a result, optimal policies in the infinite-horizon case in general also need to be randomized and history-dependent, so traditional methods are unlikely to work for the problem. This is true even for approximately optimal policies, as illustrated in the examples in Section 8.2.2. Therefore, to handle the infinite-horizon case, it is necessary to incorporate the ideas developed in our algorithm for the finite-horizon case.

Our algorithm consists of two parts. Based on the principal's discount factor $\delta_P$ and the desired accuracy $\varepsilon$, we first compute a cutoff time

$$T = O(\log(1/(\varepsilon \cdot (1 - \delta_P)))/\log(1/\delta_P)).$$

The idea is that the contribution to the overall utility after the first $T$ stages is at most $\frac{1}{1-\delta_P} \cdot \delta_P^T \leq \varepsilon$. After the $T$-th stage, we run a stationary policy that is optimal for the agent, which can be computed in polynomial time (through linear programming, or any other algorithm for computing optimal policies for standard infinite-horizon MDPs with discounted rewards). Then we treat the first $T$ stages as a finite-horizon environment, and run Algorithm 8.1 for this environment.

Since $T = O(\log(1/\varepsilon))$, this blows up the size of the problem at most by a $O(\log(1/\varepsilon))$ factor (as we make a copy of every state for every period). Two aspects of how this finite-horizon version is set up deserve mention. First, to match the infinite-horizon version, we have to discount the rewards in this finite-horizon version. This is a straightforward modification: as every state in the finite-horizon version is already indexed by time, we can simply adjust the rewards for those time-indexed states by the appropriate discount factors. Second, we still have to account for the discounted utility that the agent receives after $T$, as this may make it easier to satisfy the participation constraints before $T$. To do so, we can simply add the total expected discounted

utility after $T$ (from the agent-optimal stationary policy) as a single lump-sum reward to the final non-terminal state in the finite version. The overall policy is then to run the output of Algorithm 8.1 in the first $T$ stages, and to run the agent-optimal stationary policy after the $T$-th stage.

To see why this policy is only suboptimal by at most $\varepsilon$, observe that the expected discounted principal utility that it obtains *from the first $T$ stages* is at least the expected discounted principal utility that the overall-optimal policy obtains from those stages. This is because Algorithm 8.1 explicitly optimizes for the first $T$ stages only, and the participation constraints it faces in these first $T$ stages cannot be tighter than those faced by the optimal policy, as the participation constraints for the finite-horizon version correspond to being as generous as possible to the agent after $T$. Furthermore, the expected discounted principal utility that our algorithm obtains from the stages after $T$ can be at most $\varepsilon$ lower than that for the optimal policy, by our choice of $T$.

### 8.3.5 Proof of Theorem 8.1

In this section, we present the proof of our main result (Theorem 8.1). We start by proving several key technical lemmas. We first prove the following lemma, which provides a tractable interpretation of evaluations along specific directions.

**Lemma 8.2.** *For any state $s \in \mathcal{S}$ and direction $\alpha \in (\mathbb{R} \times \mathbb{R}_+)$,*

$$\max_{x \in D_s} \alpha \cdot (x, f_s(x)) = \max_{a \in \mathcal{A}} \left( \alpha \cdot (r_A(s,a), r_P(s,a)) + \mathbb{E}_{s' \sim P(s,a)} \left[ \max_{x' \in D_{s'} \cap \mathbb{R}_+} \alpha \cdot (x', f_{s'}(x')) \right] \right).$$

*Proof.* We first show the left hand side is greater than or equal to the right hand side. Let $a^*$ and $x_{s'} \geq 0$ for each $s' > s$ be the maximizers on the right hand side. By the definition of $f_{s'}$, each $(x_{s'}, f_{s'}(x_{s'}))$ corresponds to a subpolicy $\pi^*_{s'}$ starting from state $s'$. We have

$$(x_{s'}, f_{s'}(x_{s'})) = (u_A^{\pi^*_{s'}}(\emptyset, s'), u_P^{\pi^*_{s'}}(\emptyset, s')).$$

Moreover, for any $(h, s'') \supseteq (\emptyset, s')$, $u_A^{\pi^*_{s'}}(h, s'') \geq 0$. Now consider the policy $\pi$ defined such that $\pi(\emptyset, s) = a^*$, and for each $h = (s, a^*, s', a_2, s_3, \ldots, s_t, a_t)$ and $s'' \in \mathcal{S}$,

$$\pi(h, s'') = \pi^*_{s'}((s', a_2, s_3, \ldots, s_t, a_t), s'').$$

That is, $\pi$ follows the recommendations of $\pi^*_{s'}$ whenever the first state reached after leaving $s$ is $s'$. For any unspecified history-state pair, $\pi$ always maximizes the agent's utility. It is easy to show that

$$(u_A^\pi(\emptyset, s), u_P^\pi(\emptyset, s)) = (r_A(s, a^*), r_P(s, a^*)) + \mathbb{E}_{s' \sim P(s,a)}[(x_{s'}, f_{s'}(x_{s'}))].$$

And moreover, because each $\pi^*_{s'}$ is feasible in the future and $x_{s'} \geq 0$, $u_A^\pi(h, s'') \geq 0$ for any $(h, s'') \supseteq (\emptyset, s)$. This means

$$\begin{aligned} \max_{x \in D_s} \alpha \cdot (x, f_s(x)) &\geq \alpha \cdot (u_A^\pi(\emptyset, s), u_P^\pi(\emptyset, s)) \\ &= \alpha \cdot (r_A(s, a^*), r_P(s, a^*)) + \alpha \cdot \mathbb{E}_{s' \sim P(s,a)}[(x_{s'}, f_{s'}(x_{s'}))]. \end{aligned}$$

Now consider the other direction. Let $x^*$ be the maximizer on the left hand side, and $\pi^*$ be the corresponding policy. Without loss of generality, $\pi^*(\emptyset, s) = a^*$ is deterministic (because otherwise we can simply choose the best action in the support). For each $s'$, let $\pi_{s'}$ be such that

$$\pi_{s'}(h, s'') = \pi((s, a^*) + h, s'').$$

That is, $\pi_{s'}$ is the subpolicy starting from $s'$ induced by $\pi^*$. Then because $\pi^*$ is feasible in the future, each $\pi_{s'}$ is also feasible in the future, and moreover, $r_A^{\pi_{s'}}(\emptyset, s') \geq 0$. So we have:

$$\begin{aligned}\alpha \cdot (x^*, f_s(x^*)) &= \alpha \cdot (r_A(s, a^*), r_P(s, a^*)) + \mathbb{E}_{s' \sim P(s,a^*)}[\alpha \cdot (r_A^{\pi_{s'}}(\emptyset, s'), r_P^{\pi_{s'}}(\emptyset, s'))] \\ &\leq \max_{a \in \mathcal{A}} \left( \alpha \cdot (r_A(s, a), r_P(s, a)) + \mathbb{E}_{s' \sim P(s,a)} \left[ \max_{x' \in D_{s'} \cap \mathbb{R}_+} \alpha \cdot (x', f_{s'}(x')) \right] \right).\end{aligned}$$

This concludes the proof. □

The next lemma states that the denominators of the $x$ and $y$ coordinates returned by any evaluation of any Pareto curve throughout the algorithm are never too large, which is useful for upper bounding the number of iterations of binary search.

**Lemma 8.3.** *Consider all coordinates as fractions. Then we have: (1) the least common denominator of the $x$-coordinates of the turning points on $\{f_s\}_{s \in \mathcal{S}}$ is at most $2^{nL}$, and (2) the least common denominator of both the $x$-coordinates and the $y$-coordinates of the turning points on $\{f_s\}_{s \in \mathcal{S}}$ is at most $2^{3n^2 L}$.*

*Proof.* We start by proving the first statement by mathematical induction. For $s_{\text{term}} = n$, each turning point on $f_{s_{\text{term}}}$ is $(r_A(s_{\text{term}}, a), r_P(s_{\text{term}}, a))$ for some $a \in \mathcal{A}$, and since each $r_A(s_{\text{term}}, a)$ has at most $L$ bits, $2^L$ is a denominator of the $x$-coordinate of each turning point.

Now fix some $s < s_{\text{term}} = n$ and suppose for any $s' > s$ and any turning point on $f_{s'}$, $2^{(n-s)L}$ is a denominator of the $x$-coordinate of that point. We argue that for any turning point on $f_s$, $2^{(n-s+1)L}$ is a denominator of the $x$-coordinate of the point. Consider any turning point $(x, f_s(x))$. Observe that there is a direction $\alpha \in \mathbb{R} \times \mathbb{R}_+$ such that

$$\alpha \cdot (x, f_s(x)) = \max_{x' \in D_s} \alpha \cdot (x', f_s(x')).$$

So by Lemma 8.2, there exists an action $a \in \mathcal{A}$ and some $x_{s'} \in D_s \cap \mathbb{R}_+$, such that

$$x = r_A(s, a) + \sum_{s' > s} P(s, a, s') \cdot x_{s'}.$$

Moreover, since $x_{s'}$ is a maximizer, without loss of generality, either $x_{s'} = 0$ or $x_{s'}$ is a turning point on $f_{s'}$. In both cases, by the induction hypothesis, $2^{(n-s)L}$ is a denominator of $x_{s'}$. Since $2^L$ is a denominator of both $r_A(s, a)$ and $P(s, a, s')$, $2^{(n-s+1)L}$ must be a denominator of $x$. This establishes the first half of the lemma.

Now consider the second statement. We inductively show that for any state $s \in \mathcal{S}$, we can use $2^{3(n-s+1)nL}$ to upper bound some common denominator of both coordinates of all points on $f_{s'}$, as

well as $f_{s'}(0)$, for all $s' \geq s$. Given the first half of the lemma, we only need to argue about the $y$-coordinates.

First consider $s_{\text{term}} = n$. For the turning points, each $r_P(s_{\text{term}}, a)$ has at most $L$ bits, and $2^L$ is a denominator. As for $f_{s_{\text{term}}}(0)$, let $(x_-, y_-)$ and $(x_+, y_+)$ be the endpoints of the piece containing $(0, f_{s_{\text{term}}}(0))$ on $f_{s_{\text{term}}}$. Observe that

$$f_{s_{\text{term}}}(0) = \frac{y_- \cdot x_+ - y_+ \cdot x_-}{x_+ - x_-}.$$

So the product of the denominator of $y_- \cdot x_+ - y_+ \cdot x_-$ and the numerator of $x_+ - x_-$ is a denominator of $f_{s_{\text{term}}}(0)$. The former is at most $2^{2L}$, and the latter is at most $2 \times 2^L$, so something no larger than $2^{3L+1} \leq 2^{3nL}$ is a common denominator of all the $y$-coordinates.

Now suppose for all $s' > s$, some $D \leq 2^{3(n-s)nL}$ is a denominator of all the $y$-coordinates used to represent all $f_{s'}$ (including all turning points and $f_{s'}(0)$). We first argue that some $2^L \cdot D$ is a common denominator of all the $y$-coordinates used to represent all $f_{s'}$ for all $s' \geq s$, excluding $f_s(0)$ (we will handle $f_s(0)$ separately). Fix a turning point $(x, f_s(x))$, and again consider a direction $\alpha \in \mathbb{R} \times \mathbb{R}_+$ such that $(x, f_s(x))$ is a maximizer. By Lemma 8.2, there exists $a \in \mathcal{A}$ and $x_{s'} \in D_s \cap \mathbb{R}_+$ such that

$$f_s(x) = r_P(s, a) + \sum_{s' > s} P(s, a, s') \cdot f_{s'}(x_{s'}).$$

And each $x_{s'}$ is either a turning point or $0$. By the induction hypothesis, $2^L \cdot D$ is a denominator of all $f_s(x)$ where $(x, f_s(x))$ is a turning point. Finally consider $f_s(0)$. Again, let $(x_-, y_-)$ and $(x_+, y_+)$ be the endpoints of the piece containing $(0, f_s(0))$ on $f_s$. Observe that

$$f_s(0) = \frac{y_- \cdot x_+ - y_+ \cdot x_-}{x_+ - x_-}.$$

So the product of the denominator of $y_- \cdot x_+ - y_+ \cdot x_-$ and the numerator of $x_+ - x_-$ is a denominator of $f_s(0)$. The former, as discussed above, is at most $2^{nL} \cdot 2^L \cdot D \leq 2^{L+nL+3(n-s)nL}$, and the latter is at most $2n \times 2^{nL} \leq 2^{nL+n}$ (because the denominator of $x_+ - x_-$ is at most $2^{nL}$, and $x_+ - x_- \leq 2n$), so there exists a number that is at most $2^{3(n-s)nL+2nL+n+L} \leq 2^{3(n-s+1)nL}$ as a common denominator of all the $y$-coordinates that we care about. This finishes the proof of the lemma. □

Now we are ready to prove the correctness of Algorithm 8.1.

*Proof of Theorem 8.1.* As discussed in the overview in Section 8.3.1, Algorithm 8.1 runs in time $\text{poly}(n, m, L)$. We focus on proving the correctness of Algorithm 8.1.

In particular, for each pair $(s, \alpha)$ reached in the execution of the algorithm, $(x_{s,\alpha}, y_{s,\alpha})$ satisfies

$$\alpha \cdot (x_{s,\alpha}, y_{s,\alpha}) = \max_{x \in D_s \cap \mathbb{R}_+} \alpha \cdot (x, f_s(x)).$$

Moreover, for each $s \in \mathcal{S}$, $y_s = f_s(0)$. The claim regarding $(x_{s,\alpha}, y_{s,\alpha})$ can be proved inductively. In particular, for those points computed in the inner loop (lines 7 and 9) where $s' > s$, the property of $(x_{s',\alpha}, y_{s',\alpha})$ follows from the same property of each $(x_{s'',\alpha}, y_{s'',\alpha})$ and Lemma 8.2. As for

$(x_{s,\alpha}, y_{s,\alpha})$, the only difference is that when it is first computed in line 7, it is possible that $x_{s,\alpha} < 0$. However, this is fixed in line 15 given that $y_s = f_s(0)$.

To show $y_s = f_s(0)$, we only need to show that the binary search is accurate enough. In particular, $(x_{s,-}, y_{s,-})$ and $(x_{s,+}, y_{s,+})$ are in fact the two endpoints of the piece containing $(0, f_s(0))$. Suppose that this is not the case. That is, without loss of generality, suppose there exists a turning point $(x, y)$ to the right of $(x_{s,-}, y_{s,-})$ where $x \leq 0$. Let $\alpha = (t, 1)$ be a direction for which $(x, y)$ is the maximizer. It must be the case that $\alpha_{s,+}$ is to the right of $\alpha$, which is to the right of $\alpha_{s,-}$. In other words, at line 11, it must be the case that $\ell < t < r$. Consider the slopes of the piece containing $(0, f_s(0))$, and the piece immediately to the left of that piece, and let $k_1$ and $k_2$ be the two slopes respectively where $k_1 > k_2$. We must have $-r \leq k_2 \leq -t \leq k_1 \leq -\ell$, which in particular implies that $r - \ell \geq k_1 - k_2$. Now by Lemma 8.3, the least common denominators of the two coordinates of all turning points are at most $2^{nL}$ and $2^{3n^2L}$ respectively. Moreover, all $x$-coordinates are between $-n$ and $n$. So, the minimum possible difference between the slopes of two consecutive pieces is at least $1/(2n \cdot 2^{nL} \cdot 2^{3n^2L}) \geq 2^{-5n^2L}$. This means $r - \ell \geq k_1 - k_2 \geq 2^{-5n^2L}$, which contradicts the stopping criterion of the binary search (line 3).

One final concern is that the initial $r$ (line 2) may not be large enough. But this is impossible, because the smallest slope (which is negative) that we need to consider is $-2n \cdot 2^{nL} > -2^{3nL}$, so the initial $r = 2^{3nL}$ is in fact large enough. □

## 8.4 Future Research

In a followup paper [119], we significantly generalize the techniques presented in this chapter, and develop polynomial-time algorithms for (approximately) computing (Stackelberg) extensive-form correlated equilibria in turn-taking stochastic games. This further illustrates the power of the technical framework, which may trigger even more progress.

Throughout, we have considered a setting where the only decision the agent is able to make is to quit, and the decision to quit is irreversible. As we argued at the outset, the case where the agent only decides whether to *enter* (and this decision is irreversible) leads to the same problem. However, we could consider richer models where an agent is able to quit, but then has an opportunity to re-enter at certain later times, under certain conditions.

We have also assumed throughout that the agent has no private information. If the agent has private information, for example about how the agent values different outcomes, we arrive in a dynamic mechanism design context. As mentioned earlier, in general, in this context we face NP-hardness results [80, 115]. Still, we may ask whether the techniques developed in this chapter can be generalized to that context, perhaps resulting in polynomial-time algorithms for special cases to which the NP-hardness results do not apply.

One aspect of our approximation of the discounted infinite-horizon case is that it explicitly optimizes only for the first $T$ rounds, and consequently, it might, for example, unsustainably use up all the world's resources by round $T$. Formally, this is not a problem because, due to the nature of exponential discounting, the remaining rounds are simply not worth much. Still, one may wonder whether this fails to value long-term sustainability appropriately. Of course, this issue is not at all unique to our specific setting, but rather a fundamental aspect of exponential discounting.

**Algorithm 8.1:** A polynomial-time algorithm for computing a principal-optimal policy subject to participation constraints.

**Input:** state space $\mathcal{S} = [n]$, action space $\mathcal{A}$, reward functions $r_P$ and $r_A$, and transition probabilities $P$.

**Output:** an implicit representation of a principal-optimal policy subject to participation constraints.

/* the outer loop */

**for** $s = n, n-1, \ldots, 1$ **do**

  /* binary search for the endpoints of the piece containing $(0, f_s(0))$ */

  let $\ell \leftarrow 0$, $r \leftarrow 2^{3nL}$;

  **while** $r - \ell \geq 2^{-5n^2 L}$ **do**

    let $\alpha \leftarrow ((r+\ell)/2, 1) \in \mathbb{R}^2$ ;

    /* the inner loop */

    **for** $s' = n, n-1, \ldots, s$ **do**

      let $a_{s',\alpha} \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} \alpha \cdot \big((r_A(s',a), r_P(s',a)) + \mathbb{E}_{s'' \sim P(s',a)}[(x_{s'',\alpha}, y_{s'',\alpha})]\big)$ ;

      let $(x_{s',\alpha}, y_{s',\alpha}) \leftarrow (r_A(s', a_{s',\alpha}), r_P(s', a_{s',\alpha})) + \mathbb{E}_{s'' \sim P(s', a_{s',\alpha})}[(x_{s'',\alpha}, y_{s'',\alpha})]$ ;

      /* replace $(x_{s',\alpha}, y_{s',\alpha})$ with $(0, f_{s'}(0))$ if $x_{s',\alpha} < 0$; this is possible only for $s' > s$, where $f_{s'}(0) = y_{s'}$ has already been computed */

      **if** $x_{s',\alpha} < 0$ *and* $s' > s$ **then**

        let $(x_{s',\alpha}, y_{s',\alpha}) \leftarrow (0, y_{s'})$;

      **end**

    **end**

    let $\ell \leftarrow \alpha$ if $x_{s,\alpha} \leq 0$, and $r \leftarrow \alpha$ otherwise;

  **end**

  let $\alpha_{s,-} \leftarrow (\ell, 1)$, $\alpha_{s,+} \leftarrow (r, 1)$;

  let $(x_{s,-}, y_{s,-}) \leftarrow (x_{s,\alpha_{s,-}}, y_{s,\alpha_{s,-}})$, $(x_{s,+}, y_{s,+}) \leftarrow (x_{s,\alpha_{s,+}}, y_{s,\alpha_{s,+}})$;

  /* compute $y_s = f_s(0)$ as a linear combination of $y_{s,-}$ and $y_{s,+}$ */

  let $y_s \leftarrow (x_{s,+} \cdot y_{s,-} - x_{s,-} \cdot y_{s,+})/(x_{s,+} - x_{s,-})$;

  /* fix infeasible points reached during the binary search */

  **for** *each* $\alpha$ *tried in the above binary search where* $x_{s,\alpha} < 0$ **do**

    let $(x_{s,\alpha}, y_{s,\alpha}) \leftarrow (0, y_s)$;

  **end**

**end**

let $e_y = (0, 1)$;

**for** $s = n, n-1, \ldots, 1$ **do**

  let $a_{s,e_y} \leftarrow \operatorname{argmax}_{a \in \mathcal{A}} e_y \cdot \big((r_A(s,a), r_P(s,a)) + \mathbb{E}_{s' \sim P(s,a)}[(x_{s',e_y}, y_{s',e_y})]\big)$;

  let $(x_{s,e_y}, y_{s,e_y}) \leftarrow (r_A(s, a_{s,e_y}), r_P(s, a_{s,e_y})) + \mathbb{E}_{s' \sim P(s,a)}[(x_{s',e_y}, y_{s',e_y})]$;

  /* this time we do not need to handle $s_{\text{init}} = 1$ separately */

  **if** $x_{s,e_y} < 0$ **then**

    let $(x_{s,e_y}, y_{s,e_y}) \leftarrow (0, y_s)$;

  **end**

**end**

/* the principal's optimal reward is $y_{s_{\text{init}},e_y} = y_{1,e_y}$ */

**return** all $\{(x_{s,-}, x_{s,+})\}$, $\{x_{s,\alpha}\}$, $\{(\alpha_{s,-}, \alpha_{s,+})\}$, and $\{a_{s,\alpha}\}$ computed above;

**Algorithm 8.2:** An polynomial-time algorithm for decoding and executing (one step of) the optimal policy found by Algorithm 8.1.

**Input:** the output of Algorithm 8.1 and a history-state pair $(h, s)$ where $h = (s_1, a_1, \ldots, s_t, a_t)$.

**Output:** a possibly random action corresponding to the optimal policy found by Algorithm 8.1.

let $\alpha \leftarrow e_y = (0, 1)$;
/* trace the history and compute the current internal state of the policy */
**for** $i = 1, 2, \ldots, t$ **do**
  **if** $x_{s_i,\alpha} = 0$ **then**
    if $a_i = a_{s_i,\alpha_{s_i,-}}$ then let $\alpha \leftarrow \alpha_{s_i,-}$; otherwise let $\alpha \leftarrow \alpha_{s_i,+}$;
  **end**
**end**
**if** $x_{s,\alpha} = 0$ **then**
  /* optimal onward policy corresponds to point $(0, f_s(0))$, which requires randomization in state $s$ */
  **return** $a_{s,\alpha_{s,-}}$ with probability $x_{s,+}/(x_{s,+} - x_{s,-})$, and $a_{s,\alpha_{s,+}}$ with probability $-x_{s,-}/(x_{s,+} - x_{s,-})$;
**end**
**else**
  /* optimal onward policy corresponds to the maximizer along direction $\alpha$ */
  **return** $a_{s,\alpha}$;
**end**

# Chapter 9

# Automated Dynamic Mechanism Design

## 9.1 Introduction

In Chapter 8, we have studied a dynamic decision-making problem where the only strategic action that the agent is capable of is quitting — and importantly, the agent does not have any private information. Now let us consider a more general (and harder) decision-making problem, where the agent does have private information and can choose to misreport such information in order to maximize their own utility. This gives us a planning problem with incentive-compatibility constraints, in addition to individual rationality constraints considered in Chapter 8.

For concreteness, consider the following scenario. A company assembles an internal research group to develop key technologies to be used in the company's next-generation product in 5 years. The more progress the group makes, the more successful the product is likely to be. Since research progress is hard to monitor, the company manages the group based on its annual reports. At the beginning of each year, the group submits a report, summarizing its progress in the preceding year, as well as its needs for the current year. Taking into consideration this report (and possibly also reports from previous years), the company then decides the compensation level and the headcount of the group in the current year. Moreover, after the product launches, the company may also pay a bonus to members of the group, depending on how successful the product is.

For simplicity, suppose an annual report consists of two items: research progress (satisfactory/unsatisfactory), and need to expand (no request/request for an intern/request for a full-time employee). The company's goal is to encourage and facilitate research progress while keeping the expenses reasonable. So, a natural managing strategy is to increase (resp. decrease) the compensation level when the reported research progress is satisfactory (resp. unsatisfactory), and to allow the group to expand only when necessary, i.e., when the reported research progress is unsatisfactory. However, the research group may have a different goal than the company's. Suppose members of the group do not care about the success of the product *per se*. Instead, their primary goal is to maximize the total compensation received from the company, and for this reason, they may be incentivized to *misreport* the situation. In other words, the company faces a *dynamic mechanism design* problem, where the *principal* (i.e., the company) needs to implement (and commit to) a *mechanism* (i.e., a managing strategy) that achieves its goal through *repeated* interactions, in the

presence of strategic behavior of the *agent* (i.e., the research group).

Indeed this problem is nontrivial. For example, if the company implements the above strategy, then the group will report satisfactory progress regardless of the actual situation, which maximizes the group's total compensation over the 5 years, but also causes greater expenses for the company and jeopardizes the success of the product. To counter this, the company may additionally promise a significant bonus contingent on the success of the product. This creates incentives for the group to make more progress, and discourages overreporting the progress, because the group is not allowed to expand when the reported progress is satisfactory. That is, if actual progress is unsatisfactory, this introduces an incentive to report this truthfully so that the group may expand. However, this also runs the risk of encouraging the group to report unsatisfactory progress in order to expand even if actual progress is satisfactory, because more members always make more progress, which leads to a higher (chance of) bonus, whereas the cost of expanding is paid by the company and therefore irrelevant to the group.

One may try to fix this by introducing more rules, possibly replacing existing ones. For example, the company may allow the group to recruit an intern, but not a full-time employee, when the reported progress is unsatisfactory. Then, in the next year, if the reported progress improves, the company allows the group to make a return offer to the intern as a full-time employee. Or alternatively, the company may unconditionally allow the group to recruit interns (which are less costly), but never full-time employees. In addition to the above, the company could also temporarily decrease the compensation level when a new member joins, and later adjust the compensation based on how the reported progress improves. While all these ad hoc rules make intuitive sense, it is not immediately clear which (combinations of) rules are better, how to optimize parameters of these rules (e.g., the number of new members allowed per year and the amount by which the compensation is adjusted), or whether there is a better set of rules that look totally different.

As demonstrated by the foregoing discussion, in general, the problem of finding an optimal mechanism in *unstructured* dynamic environments, such as the above example, turns out to be extremely rich and challenging. In such environments, the actions of the principal may go beyond the allocation of items to the agent, and affect the state of the world in arbitrary ways. Moreover, both the principal and the agent may have arbitrary valuations for these actions, which also depend on the current state of the world. In economic theory, the *characterize-and-solve* approach [30, 76, 83] to mechanism design has achieved spectacular success in both static and dynamic environments, by exploiting structure of the environment to construct a characterization of optimal mechanisms, often leading to closed-form or computationally tractable solutions. However, since the environments under consideration here are loosely structured at best, the classical characterize-and-solve approach does not seem particularly suited. When disregarding the agent's incentives, one could treat the problem of finding an optimal strategy as a *planning* problem, which is known to be solvable efficiently [11, 56, 86]. However, as discussed above, the agent's strategic behavior can ruin the performance of such a strategy. From a computational perspective, while numerous methods for *automated mechanism design*, which efficiently compute optimal mechanisms without heavily exploiting structures of the environment, have been proposed [24, 25, 26], all existing methods work only for static environments with one-time interactions, and it is not immediately clear how to generalize these methods to dynamic environments. All this brings us to the following question:

*Can we efficiently compute optimal mechanisms in unstructured dynamic environments?*

### 9.1.1 Our Results

In this chapter, we study the problem of computing optimal mechanisms in *single-agent*, *discrete-time* dynamic environments with a *finite time horizon*, without any further structural assumptions. Our main results (presented in Section 9.3) can be summarized as follows:

- **Efficient algorithm**: when the time horizon is fixed, there is a polynomial-time algorithm for computing optimal mechanisms, with or without payments, that maximize the principal's utility facing a strategic agent.
- **Inapproximability**: when the time horizon can be large, it is NP-hard to approximate the principal's optimal utility within a factor of $(7/8 + \varepsilon)$ for any $\varepsilon > 0$.

To the best of our knowledge, our algorithm for constant time horizons is the first that efficiently computes optimal mechanisms in unstructured dynamic environments. The fact that our algorithm cannot scale beyond constant time horizons is by no means surprising: optimal dynamic mechanisms generally depend on the entire history, and as a result, the straightforward description of such a mechanism is exponentially large in the time horizon. Our inapproximability result further rules out the possibility of computing succinct representations of approximately optimal mechanisms that can be efficiently evaluated. These results together paint a complete picture of the computational complexity of dynamic mechanism design in unstructured environments.

### 9.1.2 Further Related Work

**Dynamic mechanism design.** The problem we study can be situated in the broad area of dynamic mechanism design, and below we discuss some representative related work. For a more comprehensive exposition, see, e.g., the survey by Pavan [82] and the one by Bergemann and Välimäki [13]. In the context of efficient (i.e., welfare-maximizing) mechanisms, Bergemann and Välimäki [12] propose the dynamic pivot mechanism, which generalizes the VCG mechanism in static environments, and Athey and Segal [4] propose the team mechanism, which focuses on budget-balancedness. As for optimal (i.e., revenue-maximizing) mechanisms, which are more closely related to our results, following earlier work [9, 30, 41], Pavan et al. [83] generalize the classical characterization by Myerson [76] into dynamic environments, unifying previous results with continuous type spaces. Ashlagi et al. [3] study ex-post individual-rational dynamic mechanisms for repeated auctions, and give an efficient $(1 - \varepsilon)$-approximation to the optimal revenue for a single agent with independent valuations across items. Mirrokni et al. [74] study non-clairvoyant dynamic mechanism design, where future distributional knowledge is unavailable to the principal. All these results for optimal mechanisms follow the characterize-and-solve approach, which is quite different from the computational approach that we take.

Particularly related to our results is the work by Papadimitriou et al. [80], who study a setting where one item is sold at each time, and agents' valuations can be correlated across items. They

show that designing an optimal deterministic mechanism is computationally hard even when there is only one agent and two items (thereby ruling out the possibility of efficiently computing optimal deterministic mechanisms in our model, which is more general). And moreover, they give a polynomial-size linear program formulation for optimal randomized mechanisms for independent agents when the number of agents and the time horizon are both constant. Restricted to a single agent, their LP formulation can be viewed as a special case of our main result: they focus on revenue maximization with a single item to be allocated at each time, in a model where the principal's actions cannot affect the future valuations of the agent; on the other hand, we allow the principal to care about actions as well as revenue, with actions being general and unstructured (as opposed to allocation/no allocation), where the future state of the world can depend arbitrarily on the principal's actions as well as the current state.

**Automated Mechanism Design.** There is a rich body of research regarding automated mechanism design (AMD) in (essentially) static environments. Conitzer and Sandholm [24, 25] initiated the study of automated mechanism design. They consider various specific static setups, and show that computing optimal deterministic mechanisms, even with a single agent, is often NP-hard (which also rules out the possibility of efficiently computing optimal deterministic mechanisms in our model, since the 1-period case is a special case), while computing optimal randomized mechanisms is often tractable. Conceptually related to our model, Hajiaghayi et al. [53] consider a model where agents enter and leave the mechanism online (but still have one-time interactions with the mechanism), and provide an algorithm for computing mechanisms that are competitive against the optimal allocation from hindsight. Sandholm et al. [91] study automated design of multistage mechanisms, but these are not for dynamic settings; instead, the motivation is to implement static mechanisms using multiple rounds of queries in order to minimize the communication cost. Sandholm and Likhodedov [90] study automated design of combinatorial auction mechanisms, and Balcan et al. [7, 8] study the sample complexity thereof. Kephart and Conitzer [64, 65] and Zhang et al. [115] (Chapter 3 of this dissertation) study AMD with partial verification and/or reporting costs. More recently, various methods have been proposed for automated mechanism design via machine learning [38, 79], and in particular, deep learning [39, 43, 87, 94]. All these results are essentially for static environments, whereas in this chapter, we focus solely on AMD in dynamic environments. Another emerging research direction is Bayesian persuasion in dynamic environments [40, 88]. In particular, Celli et al. [16] study an algorithmic persuasion problem in extensive-form games, where a single signal is sent at the very beginning, and Gan et al. [46] study an algorithmic persuasion problem in infinite-horizon discounted MDPs, where a new signal is sent at every time. These persuasion problems can be viewed as a dual problem of ours: in our problem, the principal has the commitment power, and tries to encourage the agent to truthfully report their private information, whereas in (dynamic) Bayesian persuasion, the agent has the commitment power, and tries to induce the principal to act in favor of the agent by selectively revealing their private information.

## 9.2 Preliminaries

**Dynamic environments.** Throughout this chapter, we consider single-agent, discrete-time environments with a finite time horizon. Below, we give a general definition of such a dynamic environment. Let $T$ be the time horizon, $\mathcal{S}$ be the state space, and $\mathcal{A}$ be the action space. The agent observes the state, but the principal controls the action that is taken. For each $t \in [T]$, let $v_t^P : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be the principal's valuation function, where for each state $s \in \mathcal{S}$ and action $a \in \mathcal{A}$, $v_t^P(s, a)$ is the value of the principal when playing action $a$ in state $s$, at time $t$; similarly, let $v_t^A : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ be the agent's value function. Let $P_0 \in \Delta(\mathcal{S})$ be the initial distribution over states, and for each $s \in \mathcal{S}$, denote by $P_0(s)$ the probability that the initial state is $s$. Moreover, for each $t \in [T]$, let $P_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ be the transition operator, which maps a state-action pair $(s, a)$ at time $t$ to the distribution of the next state at time $t + 1$, $P_t(s, a) \in \Delta(\mathcal{S})$. We denote by $P_t(s, a, s')$ the probability that the next state is $s'$ when playing action $a$ in state $s$ at time $t \in [T]$. For notational simplicity, let $P_0(s, a, s') = P_0(s')$ for all $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. (Note that the first *actual* action is taken at $t = 1$ — not $t = 0$ — possibly based on the state at $t = 1$.)

**Histories.** A $t$-step history is a sequence of states and actions $(s_1, a_1, s_2, \ldots, a_{t-1}, s_t, a_t)$, where for each $i \in [t]$, it is the case that $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$. For each $t \in [T]$, let $\mathcal{H}_t$ be the set of all possible $t$-step histories, i.e.,

$$\mathcal{H}_t = \{(s_1, a_1, \ldots, s_t, a_t) \mid s_i \in \mathcal{S}, a_i \in \mathcal{A} \text{ for all } i \in [t]\}.$$

For each $h = (s_1, a_1, \ldots, s_t, a_t) \in \mathcal{H}$, let $|h| = t$, and moreover, for any $s_{t+1} \in \mathcal{S}$, $a_{t+1} \in \mathcal{A}$, let $h + (s_{t+1}, a_{t+1}) = (s_1, a_1, \ldots, s_{t+1}, a_{t+1})$. Let $\mathcal{H}_0 = \{\emptyset\}$, where $\emptyset$ corresponds to the empty history with $|\emptyset| = 0$. Let $\mathcal{H} = \mathcal{H}_0 \cup \bigcup_{t \in [T-1]} \mathcal{H}_t$ be the set of all possible histories of length at most $T - 1$ in the dynamic environment. Note that $\mathcal{H}$ does not contain histories of length $T$.

**Dynamic mechanisms.** Dynamic mechanisms are more powerful than static ones, in that they may depend on the *entire history*, rather than only the current state. A (randomized) dynamic mechanism $M = (\pi, p)$ consists of an action policy $\pi$ and a payment function $p$. The action policy $\pi : \mathcal{H} \times \mathcal{S} \to \Delta(\mathcal{A})$ maps each history $h \in \mathcal{H}$, extended with the reported current state $s \in \mathcal{S}$, to a distribution over actions $\pi(h, s) \in \Delta(\mathcal{A})$. We denote by $\pi(h, s, a)$ the probability that the action taken by the mechanism is $a$ for $(h, s)$. The payment function $p : \mathcal{H} \times \mathcal{S} \to \mathbb{R}$ maps the extended history $(h, s)$ to a real number, i.e., the payment, made from the agent to the principal (but it can be negative). We remark that in principle, one can absorb payments into the action space. However, doing so would make the action space uncountable, introducing subtleties into the computational problem (which is the main focus of this chapter). Here, we keep payments separate and explicit to avoid such issues. Also, our algorithm allows linear constraints on feasible payments, including but not limited to: nonnegative payments, no payments, etc. See Section 9.3.2 for more details.

**Utilities.** Fixing a mechanism $M = (\pi, p)$, we can then define the onward utility of the principal and the agent. Let $u_P^M : \mathcal{H} \times \mathcal{S} \to \mathbb{R}$ be the principal's onward utility function under mechanism

$M$, defined inductively such that

$$u_P^M(h,s) = \sum_a \pi(h,s,a) \cdot \left( v_{|h|+1}^P(s,a) + \sum_{s'} P_{|h|+1}(s,a,s') \cdot u_P^M(h+(s,a),s') \right) + p(h,s),$$

with the boundary condition that $u_P^M(h,s) = 0$ for all $h \in \mathcal{H}_T$ and $s \in \mathcal{S}$. Here, all summations are over the entire state/action space. Let $u_P^M(\emptyset)$ be the overall utility of the principal, i.e.,

$$u_P^M(\emptyset) = \sum_s P_0(s) \cdot u_P^M(\emptyset, s).$$

Similarly, let $u_A^\pi : \mathcal{H} \times \mathcal{S} \to \mathbb{R}$ be the agent's onward utility function under mechanism $M$, defined such that

$$u_A^M(h,s) = \sum_a \pi(h,s,a) \cdot \left( v_{|h|+1}^A(s,a) + \sum_{s'} P_{|h|+1}(s,a,s') \cdot u_A^M(h+(s,a),s') \right) - p(h,s),$$

where $u_A^M(h,s) = 0$ for all $h \in \mathcal{H}_T$ and $s \in \mathcal{S}$. And let $u_A^M(\emptyset)$ be the overall utility of the agent, i.e.,

$$u_A^M(\emptyset) = \sum_s P_0(s) \cdot u_A^M(\emptyset, s).$$

We remark that while the above definition assumes that the principal cares about payments as much as the agent does, in fact, our algorithm allows for the principal to care about payments in an arbitrary linear way (including possibly not at all). See Section 9.3.2 for a detailed discussion.

**Incentive-compatible mechanisms.** We say a mechanism $M$ is incentive-compatible (IC) if the agent can never achieve a higher overall utility by misreporting the state, even in sophisticated ways. Formally, a reporting strategy $r : \mathcal{H} \times \mathcal{S} \to \mathcal{S}$ maps each history $h$ extended with the current state $s$ to a reported state $s'$, which is possibly different from $s$. Note that the agent only (mis)reports the current state, since the principal can memorize all historical reports. This reporting strategy induces a reported history $r(h) = (s'_1, a_1, \ldots, s'_t, a_t)$ for each actual history $h = (s_1, a_1, \ldots, s_t, a_t)$, where for each $i \in [t]$,

$$s'_i = r((s_1, a_1, \ldots, s_{i-1}, a_{i-1}), s_i).$$

Note that we abuse notation here: in particular, $r(h,s)$ denotes a reported state, whereas $r(h)$ denotes a reported history. And without loss of generality, we only consider deterministic reporting strategies. Given a mechanism $M$ and a reporting strategy $r$, we can define the agent's utility function $u_A^{M,r}$ under $M$ and $r$ inductively such that

$$u_A^{M,r}(h,s) = \sum_a \pi(r(h), r(h,s), a) \cdot \left( v_{|h|+1}^A(s,a) + \sum_{s'} P_{|h|+1}(s,a,s') \cdot u_A^{M,r}(h+(s,a),s') \right) - p(r(h), r(h,s)),$$

where $u_A^{M,r}(h, s) = 0$ for all $h \in \mathcal{H}_T$ and $s \in \mathcal{S}$. And let $u_A^{M,r}(\emptyset)$ be the overall utility of the agent, i.e.,

$$u_A^{M,r}(\emptyset) = \sum_s P_0(s) \cdot u_A^{M,r}(\emptyset, s).$$

In words, $u_A^{M,r}$ is the utility function of the agent applying the reporting strategy $r$ in response to the mechanism $M$. The mechanism $M$ is IC iff for any such reporting strategy $r$,

$$u_A^M(\emptyset) \geq u_A^{M,r}(\emptyset).$$

Since the revelation principle holds in dynamic environments (see, e.g., [77]), we focus on IC mechanisms in the rest of the chapter.[1]

**Individually-rational mechanisms.** When payments are allowed, it is standard to impose individual-rationality (IR) (also known as voluntary-participation) constraints on the mechanism, which roughly say that the agent should never be made worse off by participating in the mechanism. In this chapter, we consider two versions of IR constraints:

- A mechanism $M$ is overall IR if the overall utility of the agent is nonnegative, i.e., $u_A^M(\emptyset) \geq 0$. This ensures that the agent is willing to participate in the overall mechanism.
- A mechanism $M$ is dynamic IR if the onward utility of the agent is nonnegative for every history $h$ and current state $s$, i.e., $u_A^M(h, s) \geq 0$ for all $h \in \mathcal{H}$ and $s \in \mathcal{S}$. This stronger notion of IR further ensures that the agent has no incentive to leave the mechanism at any time.

As discussed in later sections, our algorithms work for all 3 cases regarding IR constraints: no IR (which results in an unbounded objective value if payments are allowed and valued by the principal), overall IR, and dynamic IR.

## 9.3 Computation of Optimal Mechanism

In this section, we investigate the computational problem of finding an optimal dynamic mechanism, which maximizes the principal's overall utility. For concreteness, we assume that all components of the dynamic environment, including the time horizon $T$, state and action spaces $\mathcal{S}$ and $\mathcal{A}$, valuation functions $v^P$ and $v^A$, and transition operator $P$, are given explicitly as input.

### 9.3.1 Hardness Result for Long-Horizon Environments

First we show that the problem with an arbitrarily large time horizon $T$ is intractable. In general, it takes exponentially many parameters in $T$ to describe a dynamic mechanism, which immediately

[1] Of course, the revelation principle will not hold in our dynamic setting if we allow it to generalize a static setting in which the revelation principle does not hold. For example, in the case of partial verification — not every type being able to misreport every other type — or costly misreporting, the revelation principle is known to hold only under certain conditions [65]. In this chapter, we only consider the standard mechanism design setting in which every type can freely misreport any other type, but our techniques can be generalized to the other settings as well.

rules out the possibility of computing a flat representation of an optimal mechanism in polynomial time. However, this leaves the possibility of computing succinct representations, e.g., an oracle which maps extended histories to distributions over actions. Our hardness result shows that it is hard to approximate the principal's maximum utility within a constant factor, which rules out the possibility of such succinct representations that can be efficiently evaluated, assuming $\mathsf{P} \neq \mathsf{NP}$. The proof of the theorem, as well as all other proofs, are deferred to the appendices.

**Theorem 9.1.** *When the time horizon $T$ can be arbitrarily large, it is $\mathsf{NP}$-hard to approximate the principal's maximum utility within a factor of $7/8 + \varepsilon$ for any $\varepsilon > 0$.*

### 9.3.2 Algorithm for Short-Horizon Environments

Now we give a polynomial-time algorithm for computing an optimal mechanism when $T$ is a constant. Our algorithm is based on a delicate linear program (LP) formulation, which relies on the following notation and concepts.

**Feasible history-state pairs.** A history-state pair $(h, s)$, where $h = (s_1, a_1, \ldots, s_t, a_t)$, is $i$-feasible if $P_j(s_j, a_j, s_{j+1}) > 0$ for every $j \in \{i, i+1, \ldots, t-1\}$, and $P_t(s_t, a_t, s) > 0$. In other words, starting from $s_i$ and taking the actions specified in $h$, there is a positive probability that the rest of the history and the state $s$ are generated from the transition operator. We say a pair $(h, s)$ is feasible if it is 1-feasible.

**Feasible extensions.** For two history-state pairs $(h, s)$ and $(h', s')$ where $h = (s_1, a_1, \ldots, s_t, a_t)$ and $h' = (s'_1, a'_1, \ldots, s'_{t'}, a'_{t'})$, we say that $(h', s')$ feasibly extends $(h, s)$, i.e., $(h, s) \subseteq (h', s')$, if $(h, s) = (h', s')$, or the following conditions hold simultaneously:

- $t = |h| < |h'| = t'$.
- For any $i \in [t]$, $(s_i, a_i) = (s'_i, a'_i)$ (this holds automatically when $h = \emptyset$ and therefore $|h| = 0$).
- $s = s'_{t+1}$.
- $(h', s')$ is $(|h| + 1)$-feasible (note that this does not require $h$ itself to be feasible).

**Extended transition operator.** For notational simplicity we define the following extended transition operator $P^E_t : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S})$ for all $t \in \{0\} \cup [T]$, such that

$$P^E_t(s, a, s') = \begin{cases} P_t(s, a, s'), & \text{if } P_t(s, a, s') > 0 \\ 1, & \text{otherwise} \end{cases}.$$

In words, the extended transition operator assigns phantom probability 1 to each way of transitioning that happens with probability 0 (so $P^E_t(s, a)$ does not always normalize to 1). As a shorthand, let $P^E_0(s') = P^E_0(s, a, s')$ for some $s \in \mathcal{S}$ and $a \in \mathcal{A}$ (the specific choice does not matter). The extended transition operator helps in constructing the flow and IC constraints below and simplifies the formulation. In particular, we always have $P^E_t(s, a, s') > 0$.

**Last state-action pair.** For a history $h \in \mathcal{H}$ where $h = (s_1, a_1, \ldots, s_t, a_t)$, we use $\text{last}(h)$ as a shorthand for the last state-action pair, i.e., $\text{last}(h) = (s_t, a_t)$. In particular, when $h = \emptyset$, $\text{last}(h)$ can be any state-action pair (the choice does not affect our results — it is merely a simplifying shorthand).

Now we are ready to describe the LP formulation. The complete formulation is given in Figure 9.1. The formulation is for nonnegative payments and dynamic IR constraints — we will discuss later how the formulation can be modified to allow other types of constraints. Below, we describe each of its components.

**Variables, flow constraints, and the corresponding mechanism.** There are $5$ classes of variables in the LP:

- $x(h, s, a)$: the absolute, unconditional probability that the mechanism reaches state $s$ via history $h$, and takes action $a$.
- $y(h, s)$: the payment for history-state pair $(h, s)$, scaled by the probability that the mechanism reaches $s$ via $h$ (i.e., $z(h, s)$).
- $z(h, s)$: the probability that the mechanism reaches state $s$ via history $h$, which by definition satisfies
$$z(h, s) = \sum_{a \in \mathcal{A}} x(h, s, a).$$
- $u(h, s)$: the onward utility of the agent at state $s$ with history $h$ assuming truthful reporting, scaled by the probability that the mechanism reaches $s$ via $h$ (i.e., $z(h, s)$).
- $u(h, s, s')$: the onward utility of the agent at state $s$ with history $h$ if the agent misreports $s'$, assuming truthful reporting in the future, scaled by the probability that the mechanism reaches $s'$ via $h$ (i.e., $z(h, s')$).

The flow constraints (Eq. (9.2)-(9.4)) enforce roughly the above interpretation of variables to $x(h, s, a)$ and $z(h, s)$, except for ways of transition that have probability $0$. For each way of transition with probability $0$, the extended transition operator assigns phantom probability $1$. This phantom probability is not counted in the objective function (because only feasible history-state pairs are counted) or in the utility variables $u(h, s)$ (because only feasible extensions are counted). So, the phantom probability does not affect the principal's or the agent's utility assuming truthful reporting. Instead, together with other constraints, it guarantees that the mechanism behaves well even for history-state pairs that appear with probability $0$ under truthful reporting, which is necessary for the mechanism to be IC (see later paragraphs). Under the above interpretation, the LP variables (and in particular, $x(h, s, a)$, $y(h, s)$ and $z(h, s)$) naturally correspond to a mechanism $M = (\pi, p)$. Formally, for each $h \in \mathcal{H}$, $s \in \mathcal{S}$:

- If $z(h, s) > 0$, then
$$p(h, s) = y(h, s)/z(h, s),$$
and for each $a \in \mathcal{A}$,
$$\pi(h, s, a) = x(h, s, a)/z(h, s).$$
- If $z(h, s) = 0$, then let $\pi(h, s)$ be an arbitrary distribution over $\mathcal{A}$, and $p(h, s) = 0$.

The feasibility of the mechanism (i.e., every $\pi(h, s)$ is a distribution over $\mathcal{A}$ and every $p(h, s)$ is nonnegative) is guaranteed by constraints (9.2), (9.9) and (9.10). We remark that while the mechanism constructed from the LP variables may not be unique, effectively this makes no difference, since the parts of the mechanism that are chosen arbitrarily can never be accessed when executing the mechanism. This is because $z(h, s) = 0$ only if at some point in the history $h$, there is an action that the mechanism would never play given the reported states and actions before that. In particular, the above does not simply apply to all history-state pairs $(h, s)$ that are reached with probability $0$ under truthful reporting, in which case $z(h, s)$ may still be positive due to the extended transition operator. Moreover, given any mechanism, one can construct LP variables in a similar way, such that the mechanism constructed from these variables is the same as the original mechanism (modulo the unreachable parts). In other words, the above correspondence is effectively bijective.

**The objective.** The objective function of the LP (Eq. (9.1)) is precisely the overall utility of the principal under the mechanism constructed above, assuming truthful reporting. This is captured by the following lemma.

**Lemma 9.1.** *Let $M = (\pi, p)$ be the mechanism constructed from variables $x(h, s, a)$, $y(h, s)$, and $z(h, s)$ which satisfy the flow constraints. Then*

$$u_P^M(\emptyset) = \sum_{h \in \mathcal{H}, s \in \mathcal{S}: (h,s) \text{ is feasible}} \left( \sum_{a \in \mathcal{A}} v_{|h|+1}^P(s, a) \cdot x(h, s, a) + y(h, s) \right).$$

From this lemma, it is clear that the objective of the LP is the natural quantity to maximize.

**Utility.** The utility constraints (Eq. (9.5)) collect the agent's onward utility, where $u(h, s)$ is equal to the agent's onward utility in state $s$ from history $h$, assuming truthful reporting, scaled by $z(h, s)$. This is captured by the following lemma.

**Lemma 9.2.** *Let $M = (\pi, p)$ be the mechanism constructed from variables $x(h, s, a)$, $y(h, s)$, and $z(h, s)$ which satisfy the flow and utility constraints. For all $h \in \mathcal{H}$, $s \in \mathcal{S}$,*

$$u(h, s) = z(h, s) \cdot u_A^M(h, s).$$

The proof of Lemma 9.2 is essentially the same as that of Lemma 9.1. Given the correspondence to the agent's utility $u_A^M(h, s)$, the utility variables $u(h, s)$ act as auxiliary variables in IC constraints.

**IC constraints.** IC constraints are a key component of the LP formulation. There are two families of IC constraints: collecting the agent's scaled utility from single-step misreporting (Eq. (9.6)), and subsequently restricting the mechanism such that there is no incentive for misreporting (Eq. (9.7)). In Eq. (9.6), we build variables $u(h, s, s')$, which is supposed to be the onward utility of the agent in state $s$ from history $h$ misreporting $s'$, assuming truthful reporting in the future, scaled by $z(h, s')$ (rather than $z(h, s)$). This is captured by the following lemma.

**Lemma 9.3.** *Let $M = (\pi, p)$ be the mechanism constructed from variables $x(h, s, a)$, $y(h, s)$, and $z(h, s)$ which satisfy the flow constraints, the utility constraints, and Eq.* (9.6)*. Then the following statement holds: for all $h \in \mathcal{H}$, $s, s' \in \mathcal{S}$, let reporting strategy $r_{h,s,s'}$ be such that*

$$r_{h,s,s'}(h', s'') = \begin{cases} s', & \text{if } h = h' \text{ and } s = s'' \\ s'', & \text{otherwise} \end{cases}.$$

*That is, $r_{h,s,s'}$ misreports $s'$ only in state $s$ from history $h$, and reports truthfully otherwise. Then for all $h \in \mathcal{H}$, $s, s' \in \mathcal{S}$,*

$$u(h, s, s') = z(h, s') \cdot u_A^{M, r_{h,s,s'}}(h, s).$$

Given Lemma 9.3, Eq. (9.7) then guarantees that the mechanism $M$ is robust against single-step misreporting for all reachable history-state pairs.

**Lemma 9.4.** *Let $M = (\pi, p)$ be the mechanism constructed from variables $x(h, s, a)$, $y(h, s)$, and $z(h, s)$ which satisfy the flow constraints, the utility constraints, and Eq.* (9.6)*. The following is true if and only if the LP variables also satisfy Eq.* (9.7)*: for all $h \in \mathcal{H}$, $s, s' \in \mathcal{S}$ where $(h, s)$ is reachable by the mechanism $M$,*

$$u_A^M(h, s) \geq u_A^{M, r_{h,s,s'}}(h, s).$$

We then show that a mechanism is IC if and only if there is no incentive for single-step misreporting, which directly implies that the mechanism $M$ constructed from the LP variables is IC. This is captured by the following lemma.

**Lemma 9.5.** *Let $M = (\pi, p)$ be the mechanism constructed from variables $x(h, s, a)$, $y(h, s)$, and $z(h, s)$ which satisfy the flow constraints, the utility constraints, and Eq.* (9.6)*. Then $M$ is IC if and only if the LP variables also satisfy Eq.* (9.7)*.*

**IR constraints, feasible actions, and feasible payments.** These constraints are straightforward given the correspondence between the LP variables and the mechanism that we have discussed above. Note that while Eq. (9.8) is for dynamic IR (i.e., the agent has no incentive to leave the mechanism at any point) and Eq. (9.10) is for nonnegative payments, it is easy to replace them with similar constraints that correspond to overall IR or no payments. See Appendix C for more details.

**Optimality of LP solution.** Given the above facts, we are ready to state and prove the main result of the chapter.

**Theorem 9.2.** *There is an algorithm which computes an optimal IC and (optionally) IR dynamic mechanism, with or without payments, in time $O(\text{poly}(|\mathcal{S}|^T, |\mathcal{A}|^T, L))$, where $L$ is the number of bits required to encode each of the input parameters. In particular, when $T$ is constant, the algorithm runs in polynomial time.*

## 9.4 The Case of Myopic Agents: Characterization and Faster Algorithm

In this section, we briefly discuss a special case of the problem of computing optimal dynamic mechanisms, namely the case where the agent is myopic, or, equivalently, the agent has a discount factor of $0$. While our LP-based algorithm still applies, as we will see below, optimal mechanisms for myopic agents enjoy a succinct representation in this case, which also enables a faster algorithm that scales only linearly in the time horizon $T$. See Section 9.7 for more details, including the formal definition of myopic agents and the complete description of the algorithm.

### 9.4.1 Characterization of Optimal Mechanisms

We first show that when the agent is myopic, without loss of generality, the actions and payments specified by an optimal mechanism depend only on the time, the previous state, the previous action and the current state (we call such a mechanism a *succinct mechanism*), instead of the entire history-state pair.

**Lemma 9.6.** *Fix a dynamic environment. When the agent is myopic, for any IC mechanism $M = (\pi, p)$, there is another IC mechanism $M' = (\pi', p')$ (which is IR whenever $M$ is) such that*

- $u_P^{M'}(\emptyset) \geq u_P^M(\emptyset)$*, and*
- *for all $h \in \mathcal{H}$, $s \in \mathcal{S}$, $\pi'$ and $p'$ depend only on $|h|$, $s_p$, $a_p$ and $s$, where $(s_p, a_p) = \text{last}(h)$.*

*Moreover, the above is true regardless of whether payments are allowed, or which IR constraints are required.*

### 9.4.2 Faster Algorithm for Myopic Agents

Based on the above characterization, we present a faster algorithm for computing an optimal mechanism in the face of a myopic agent. In particular, the time complexity of this algorithm depends only linearly on the time horizon $T$, making it feasible for dynamic environments with a long time horizon. This is in contrast with the case of patient agents, for which, as we have seen, the long-horizon problem is hard to approximate. The algorithm uses as a subroutine a blackbox algorithm that computes an optimal IC (and optionally IR) mechanism in static environments, with or without payments. It is known that such an algorithm can be implemented using linear programming, and in some cases in more efficient ways [24, 26, 115].

**Theorem 9.3.** *When the agent is myopic, Algorithm 9.1 computes an optimal IC and (optionally) IR dynamic mechanism, with or without payments, in time*

$$O(T|\mathcal{S}||\mathcal{A}| \cdot T_{\text{stat}}(|\mathcal{S}|, |\mathcal{A}|, L)) = O(T \cdot \text{poly}(|\mathcal{S}|, |\mathcal{A}|, L)),$$

*where $T_{\text{stat}}$ is the time complexity of the blackbox algorithm used for computing an optimal IC (and optionally IR) mechanism in static environments, and $L$ is the number of bits required to encode each of the input parameters.*

## 9.5 Conclusion

We studied automated dynamic mechanism design and showed that, while it is computationally hard to find (even approximately) optimal mechanisms when (1) facing a patient agent and (2) the horizon is long, when either of these two conditions is dropped, an optimal mechanism can be found efficiently. An interesting future direction is to generalize our results to related problems with a stronger learning flavor, e.g., reinforcement learning with IC and/or IR constraints.

Besides using our algorithms directly for appropriate applications, the experimental results that they enable (including those that we presented in Appendix F) can guide new theory. For example, can we rigorously prove the benefit of facing a patient agent when the setting is not all too adversarial, and perhaps even characterize the transition point at which facing a patient agent becomes better than facing a myopic one? Analytically derived mechanisms can also be compared to these experimental results to see how close to optimal in performance they are. Finally, close inspection of the actual mechanisms generated by our algorithms may reveal insights that can be used to analytically design new mechanisms.

## 9.6 Customizing the LP Formulation.

The LP formulation in Figure 9.1 allows for nonnegative payments, assumes that the principal cares about payments as much as the agent, and enforces dynamic IR constraints. As mentioned above, one can customize all these components by modifying the corresponding parts of the LP formulation. Below we discuss several ways of customization.

- **Unequal valuations for payments**: in the case where the principal has utility $c$ for one unit of payment (whereas without loss of generality the agent has utility 1), one may replace the objective function (Eq. (9.1)) with

$$\max \sum_{h\in\mathcal{H},s\in\mathcal{S}:(h,s) \text{ is feasible}} \left( \sum_{a\in\mathcal{A}} v^P_{|h|+1}(s,a)\cdot x(h,s,a) + c\cdot y(h,s) \right).$$

  Note that our formulation works only when the principal cares linearly about payments. Notably, the principal may not care about payments at all (as in the case of paying the agent in "brownie points"), or even dislike payments made by the agent (as in the case where the agent is asked to expend useless effort or "burn money" and the principal cares in part about the resulting loss of welfare).

- **No payments**: to forbid payments in the mechanism, one can simply replace Eq. (9.10) with

$$y(h,s) = 0,\ \forall h \in \mathcal{H}, s \in \mathcal{S}.$$

- **Feasible intervals of payments**: more generally, one may wish to specify a feasible interval $[a_{h,s}, b_{h,s}]$ for the payment at each history-state pair $(h,s)$ such that $a_{h,s} \le p(h,s) \le b_{h,s}$, which subsumes both nonnegative payments and no payments as special cases. This can be done by replacing Eq. (9.10) with

$$a_{h,s}\cdot z(h,s) \le y(h,s) \le b_{h,s}\cdot z(h,s),\ \forall h\in\mathcal{H}, s\in\mathcal{S}.$$

- **Overall/no IR**: when the agent can choose whether to participate in the mechanism, but cannot leave halfway (corresponding to an overall IR constraint), one can replace Eq. (9.8) with

$$\sum_{s\in\mathcal{S}} u(\emptyset, s) \geq 0.$$

  Also, when leaving the mechanism is not an option for the agent from the very beginning (corresponding to no IR constraint), one may remove IR constraints simply by removing Eq. (9.8).

- **Discount factors**: to accommodate the case where the agent has a discount factor $0 \leq \delta < 1$, one can modify the LP formulation in the following way:
  - Replace Eq. (9.5) with

$$u(h,s) = \sum_{h'\in\mathcal{H}, s'\in\mathcal{S}:(h,s)\subseteq(h',s')} \delta^{|h'|-|h|} \cdot \left( \sum_{a\in\mathcal{A}} v^A_{|h'|+1}(s',a) \cdot x(h',s',a) - y(h',s') \right), \quad \forall h \in \mathcal{H}, s \in \mathcal{S}.$$

  - Replace Eq. (9.6) with

$$u(h,s,s') = \sum_{a\in\mathcal{A}} v^A_{|h|+1}(s,a) \cdot x(h,s',a) - y(h,s') + \delta \cdot \sum_{a\in\mathcal{A}, s''\in\mathcal{S}} \frac{P_{|h|+1}(s,a,s'')}{P^E_{|h|+1}(s',a,s'')} \cdot u(h+(s',a),s''),\ \forall h \in \mathcal{H}, s, s' \in \mathcal{S}$$

- **Deterministic mechanisms**: the problem of computing an optimal deterministic mechanism is NP-hard even in static environments [24, 25]. Nevertheless, given our LP formulation, one can restrict the mechanism to be deterministic by introducing Boolean variables, resulting in a mixed integer LP. While integer LPs are hard to solve in a worst-case sense, real-world problems often admit certain structures which can be exploited by commercial solvers such as CPLEX and Gurobi. To be specific, we introduce a Boolean variable $c(h,s,a)$ which controls $x(h,s,a)$ for all $h \in \mathcal{H}$, $s \in \mathcal{S}$, and $a \in \mathcal{A}$, and ensures that fixing $h$ and $s$, $x(h,s,a)$ can be positive for at most one action $a \in \mathcal{A}$. This is implemented by the following constraints (in addition to the existing ones):

$$\begin{aligned}
&x(h,s,a) \leq c(h,s,a) && \forall h \in \mathcal{H}, s \in \mathcal{S}, a \in \mathcal{A} \\
&\sum_{a\in\mathcal{A}} c(h,s,a) = 1 && \forall h \in \mathcal{H}, s \in \mathcal{S} \\
&c(h,s,a) \in \{0,1\} && \forall h \in \mathcal{H}, s \in \mathcal{S}, a \in \mathcal{A}.
\end{aligned}$$

We also remark that the above discussion is non-exhaustive: one can impose richer restrictions by modifying the LP formulation in other linear ways, and/or combining the above modifications.

## 9.7 The Case of Myopic Agents: Characterization and Faster Algorithm

In this section, we consider a special case of the problem of computing optimal dynamic mechanisms, namely the case where the agent is myopic, or, equivalently, the agent has a discount factor of $0$. While our LP-based algorithm still applies, as we will see below, optimal mechanisms for myopic agents enjoy a succinct representation in this case, which also enables a faster algorithm that scales only linearly in the time horizon $T$.

**Myopic agents.** The utility $u_A^M$ of a myopic agent under mechanism $M$ is such that

$$u_A^M(h,s) = \sum_a \pi(h,s,a) \cdot v^A_{|h|+1}(s,a) - p(h,s),$$

where $u_A^M(h,s) = 0$ for all $h \in \mathcal{H}_T$ and $s \in \mathcal{S}$. Given a reporting strategy $r$, the utility $u_A^{M,r}$ of the agent under mechanism $M$ and reporting strategy $r$ is

$$u_A^{M,r}(h,s) = \sum_a \pi(r(h), r(h,s), a) \cdot v^A_{|h|+1}(s,a) - p(r(h), r(h,s)).$$

$M$ is IC if and only if for all $h \in \mathcal{H}$ and $s \in \mathcal{S}$, there are no future reporting strategies that lead to better utility, i.e., for every reporting strategy $r$ where $r(h',s') = s'$ whenever $|h'| < |h|$,

$$u_A^M(h,s) \geq u_A^{M,r}(h,s).$$

Note that since the agent is myopic, it is insufficient to simply require $u_A^M(\emptyset) \geq u_A^{M,r}(\emptyset)$. Also, it is necessary to restrict misreporting to the future, since otherwise the agent would be allowed and sometimes incentivized to change the past, leading to unrealistically strong IC requirements. Again, since the revelation principle holds, we focus only on IC mechanisms.

### 9.7.1 Characterization of Optimal Mechanisms

We first show that when the agent is myopic, without loss of generality, the actions and payments specified by an optimal mechanism depend only on the time, the previous state, the previous action and the current state (we call such a mechanism a *succinct mechanism*), instead of the entire history-state pair.

**Lemma 9.7.** *Fix a dynamic environment. When the agent is myopic, for any IC mechanism $M = (\pi, p)$, there is another IC mechanism $M' = (\pi', p')$ (which is IR whenever $M$ is) such that*

- $u_P^{M'}(\emptyset) \geq u_P^M(\emptyset)$, *and*
- *for all $h \in \mathcal{H}$, $s \in \mathcal{S}$, $\pi'$ and $p'$ depend only on $|h|$, $s_p$, $a_p$ and $s$, where $(s_p, a_p) = \text{last}(h)$.*

*Moreover, the above is true regardless of whether payments are allowed, or which IR constraints are required.*

**Algorithm 9.1:** Algorithm for computing an optimal mechanism against a myopic agent.

**Input:** Time horizon $T$, transition probabilities $\{P_t\}_{t\in[T]}$, principal's valuation functions $\{v_t^P\}_{t\in[T]}$, agent's valuation functions $\{v_t^A\}_{t\in[T]}$.

**Output:** An optimal IC (for a myopic agent) mechanism $M = (\pi, p)$.

**for** $t = T, T-1, \ldots, 1$ **do**
  **for** $s \in \mathcal{S}$, $a \in \mathcal{A}$ **do**
    let $u(t, s, a) \leftarrow v_t^P(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') \cdot u_P^M(t+1, s, a, s')$;
    /* the above operation is well-defined, in particular because $u_P^M(t+1, s, a, s')$ depends only on the part of $M$ that has already been computed */
  **end**
  **for** $s_p \in \mathcal{S}, a_p \in \mathcal{A}$ **do**
    let $(\pi', p') \leftarrow \mathsf{OptStatMech}(\mathcal{S}, \mathcal{A}, \{P_{t-1}(s_p, a_p, s)\}_s, \{u(t, s, a)\}_{s,a}, \{v_t^A(s, a)\}_{s,a})$;
    /* call OptStatMech to compute an optimal static mechanism $(\pi', p')$, in a static environment with type space $\mathcal{S}$, action space $\mathcal{A}$, population distribution $\{P_{t-1}(s_p, a_p, s)\}_s$, principal's utility function $\{u(t, s, a)\}_{s,a}$, and agent's utility function $\{v_t^A(s, a)\}_{s,a}$ */
    **for** $s \in \mathcal{S}$ **do**
      let $\pi(t, s_p, a_p, s) \leftarrow \pi'(s)$, and $p(t, s_p, a_p, s) \leftarrow p'(s)$;
    **end**
  **end**
**end**
return $M = (\pi, p)$;

### 9.7.2 Faster Algorithm for Myopic Agents

Based on the above characterization, we present below a faster algorithm for computing an optimal mechanism in the face of a myopic agent. In particular, the time complexity of this algorithm depends only linearly on the time horizon $T$, making it feasible for dynamic environments with a long time horizon. This is in contrast with the case of patient agents, for which, as we have seen, the long-horizon problem is hard to approximate.

To improve readability, we use the following shorthand notation for succinct mechanisms. For a succinct mechanism $M = (\pi, p)$, for any $h \in \mathcal{H}$ and $s \in \mathcal{S}$, let $\pi(t, s_p, a_p, s) = \pi(h, s)$ be the action policy at $(h, s)$, and $p(t, s_p, a_p, s) = p(h, s)$ be the payment function, where $(s_p, a_p) = \mathrm{last}(h)$ and $t = |h| + 1$. Also, observe that the principal's onward utility at any history-state pair $(h, s)$ depends only on the previous state $s_p$, the previous action $a_p$, and the current state $s$. In such cases, we also denote this utility by $u_P^M(t, s_p, a_p, s) = u_P^M(h, s)$.

The full algorithm is given as Algorithm 9.1. It uses as a subroutine an algorithm OptStatMech which computes an optimal IC (and optionally IR) mechanism in static environments, with or without payments. It is known that such an algorithm can be implemented using linear programming, and in some cases in more efficient ways [24, 26, 115]. Algorithm 9.1 proceeds in an inductive

fashion, building a succinct mechanism backwards, one layer at a time. It repeatedly solves the problem of maximizing the principal's expected onward utility over the current state $s$, given the previous state $s_p$ and the previous action $a_p$. Since $s_p$ and $a_p$ together induce a roll-in distribution over the state space, this problem can be reduced to computing an optimal static mechanism, where the valuation function of the principal depends on the optimal mechanism in the following layers. This can then be solved by calling OptStatMech, the algorithm for computing an optimal static mechanism. Below we state and prove the correctness and computational efficiency of Algorithm 9.1.

**Theorem 9.4.** *When the agent is myopic, Algorithm 9.1 computes an optimal IC and (optionally) IR dynamic mechanism, with or without payments, in time*

$$O(T|\mathcal{S}||\mathcal{A}| \cdot T_{\text{stat}}(|\mathcal{S}|, |\mathcal{A}|, L)) = O(T \cdot \text{poly}(|\mathcal{S}|, |\mathcal{A}|, L)),$$

*where $T_{\text{stat}}$ is the time complexity of* OptStatMech*, and $L$ is the number of bits required to encode each of the input parameters.*

**Customizing Algorithm 9.1.** We remark that Algorithm 9.1 can also be customized to allow for unequal valuations of payments, feasible intervals of payments, etc. Moreover, it can be adapted to compute an optimal deterministic mechanism, by requiring OptStatMech to compute an optimal deterministic static mechanism. Again, while this is generally hard to compute, for practical purposes, it is reasonable to expect that OptStatMech implemented using commercial mixed integer LP solvers (or in other practically efficient ways) can find an optimal mechanism efficiently.

## 9.8 Infeasibility of Memoryless Mechanisms

From a planning perspective, automated dynamic mechanism design can be viewed equivalently as planning in MDPs where the current state cannot be directly observed, but instead, has to be reported by a strategic agent whose interest may not align with the planner's. In particular, when the planner and the agent share the same valuation function, automated dynamic mechanism design degenerates to the classical problem of planning in episodic MDPs with a finite planning horizon. In the latter problem, it is well known that without loss of generality, any optimal policy depends only on the time and the current state, i.e., it is memoryless. And moreover, such optimal policies can be computed in polynomial time. In light of the above facts, the following questions arise naturally: are there (approximately) optimal mechanisms that are also memoryless, and can we find optimal memoryless mechanisms efficiently? In this section, we give negative answers to both questions, which means memoryless mechanisms are generally infeasible for dynamic environments. We first show that memoryless mechanisms can be arbitrarily worse than general, history-dependent mechanisms, against both patient and myopic agents.

**Theorem 9.5.** *Regardless of whether the agent is myopic, for any $\varepsilon > 0$, there is a dynamic environment where the principal's utility under an optimal memoryless mechanism is at most an $\varepsilon$ fraction of the principal's optimal utility.*

Now we show that on top of the suboptimality, optimal memoryless mechanisms are computationally hard to approximate.

**Theorem 9.6.** *Regardless of whether the agent is myopic, it is* NP*-hard to approximate the principal's maximum utility under memoryless mechanisms within a factor of* $7/8 + \varepsilon$ *for any* $\varepsilon > 0$.

## 9.9 Experimental Results

In this section, we present experimental results where our algorithms are applied to synthetic dynamic environments of different characteristics. The main goals of the experiments are

- to provide a proof of concept for the methods proposed in this chapter,
- to illustrate the necessity of considering incentives when planning in dynamic environments (as opposed to disregarding the agent's valuations and treating the problem simply as an MDP based on the principal's valuations),
- to study the effect of cooperation and competition in dynamic mechanism design, and
- to understand the difference between patient and myopic agents from the principal's perspective, especially when the parameters of the environment vary.

### 9.9.1 Setup of Experiments

**Mechanisms/models of the agent under consideration.** For each dynamic environment examined, we consider the following quantities from different combinations of mechanisms and models of the agent:

- **Naïve mechanisms facing a naïve agent**: the principal's optimal utility facing a naïve agent who always reports truthfully, i.e., the optimal utility when treating the problem simply as an MDP based on the principal's valuations.
- **Naïve mechanisms facing a patient agent**: the principal's utility, when executing the optimal mechanism/policy for naïve agents, facing a strategic agent who is patient.
- **Naïve mechanisms facing a myopic agent**: the principal's utility, when executing the optimal mechanism/policy for naïve agents, facing a strategic agent who is myopic.
- **Patient mechanisms facing a patient agent**: the principal's optimal utility facing a strategic agent who is patient.
- **Myopic mechanisms facing a myopic agent**: the principal's optimal utility facing a strategic agent who is myopic.

For simplicity, payments are not allowed in any of our experiments.

**Dynamic environments.** To manifest the effect of cooperation and competition, we generate synthetic dynamic environments in the following way:

- Fix the time horizon $T$, number of states $|\mathcal{S}|$, number of actions $|\mathcal{A}|$, and correlation parameter $\eta \in [-1, 1]$ (explained below).
- Let the initial distribution $P_0$ be a random distribution generated in the following way: for each state $s$, we generate a uniformly random real number $\mathrm{rand}(s)$ between $0$ and $1$, which

is proportional to $P_0(s)$. That is, $P_0(s) = \mathrm{rand}(s) / \left(\sum_{s'} \mathrm{rand}(s')\right)$.

- For each $t \in [T]$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$, we generate the transition distribution $P_t(s, a)$ independently in the same way that $P_0$ is generated.
- For each $t \in [T]$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$, let $v_t^P(s, a)$ be an independent, uniformly random real number between $0$ and $1$.
- For each $t \in [T]$, $s \in \mathcal{S}$ and $a \in \mathcal{A}$, let $v_t^A(s, a) = \eta \cdot v_t^P(s, a) + (1 - |\eta|) \cdot \mathrm{rand}(t, s, a)$, where $\mathrm{rand}(t, s, a)$ is an independent, uniformly random real number between $0$ and $1$.

The correlation parameter $\eta$ controls the extent to which the interests of the principal and the agent are (mis)aligned. In particular, if $\eta = 1$, then the principal and the agent have exactly the same valuations, corresponding to full cooperation. If $\eta = -1$, then the principal and the agent are in a zero-sum situation, corresponding to full competition.

### 9.9.2 Summary of Experimental Results

**Suboptimality of naïve mechanisms.** As we can see from Figure 9.2, even when the state and action spaces are extremely simple, i.e., there are only $2$ states and $2$ actions, when the correlation parameter $\eta = -1$ (i.e., when the agent acts adversarially), naïve mechanisms facing a strategic agent can only achieve about $75\%$ of the naïve benchmark, i.e., the optimal utility when the agent is naïve. When $\eta = 0$ (i.e., when the agent's and principal's valuations are independent), naïve mechanisms facing a strategic agent still achieve only $85\%$ of the naïve benchmark. On the other hand, the respective optimal mechanisms facing a patient or myopic agent consistently achieve about $95\%$ of the naïve benchmark. This gap is further amplified in Figure 9.3: as the environment becomes more and more complex (i.e., the numbers of states and actions become larger and larger), the utility of naïve mechanisms facing a strategic agent drops below $20\%$ of the naïve benchmark when $\eta = -1$, and to about $50\%$ when $\eta = 0$. In contrast, the respective optimal mechanisms facing a patient or myopic agent still achieve about $70\%$ of the naïve benchmark even when $\eta = -1$. These phenomena suggest that when the agent is not fully cooperative, taking strategic behavior into consideration significantly improves the principal's utility, even in extremely simple dynamic environments. Moreover, the more complex the environment is, the larger this gap becomes.

Another interesting fact to note is that even when the principal's and the agent's valuations are exactly the same (i.e., when $\eta = 1$), naïve mechanisms are still suboptimal facing a myopic agent, since the agent may sacrifice greater long-term gain in exchange for smaller immediate value. This phenomenon is more significant in Figure 9.2, especially in environments with longer time horizons. In such cases, taking into consideration the fact that the agent is myopic mitigates the loss, and recovers almost all the utility of the naïve benchmark.

**Effect of cooperation and competition.** As the correlation parameter increases, both Figure 9.2 and Figure 9.3 show clear upward trends in all the quantities that we consider (except for the naïve benchmark which is always normalized to $1$), as one would expect. Nevertheless, we note the following facts from the figures: compared to naïve mechanisms, optimal mechanisms facing a strategic agent are much less affected by the correlation parameter. Moreover, as Figure 9.2

shows, the performance of optimal mechanisms facing a strategic agent is remarkably stable as the time horizon grows. In other words, in random dynamic environments, the utility loss caused by competing interests of the principal and the agent is only mildly amplified by long time horizons.

**Difference between patient and myopic agents.** As can be seen from the figures, regardless of whether the agent is patient or myopic, the principal's optimal utility is almost the same. Nevertheless, the difference appears to be amplified as the time horizon grows (see Figure 9.2). When the correlation parameter $\eta = -1$, the optimal utility facing a myopic agent is noticeably larger than that facing a patient agent — which makes sense as only the patient agent has interests truly opposite those of the principal. This gap shrinks as $\eta$ becomes larger, and vanishes when $\eta$ is around $-0.25$. Then, as $\eta$ continues to grow, the optimal utility facing a myopic agent falls behind and never catches up. In particular, when $\eta = 1$, the optimal utility facing a patient agent is the same as the naïve benchmark, whereas that facing a myopic agent is slightly smaller. The above phenomena indicate that in environments with a long time horizon, myopic agents are easier to exploit, while patient agents are easier to cooperate with. Interestingly, the critical value of $\eta$, where the optimal utility facing a patient agent catches up, is about $-0.25$ instead of $0$, which suggests that even when the principal's and the agent's valuations are mildly negatively correlated, it is possible to find a middle ground where cooperation is more beneficial than exploitation in the long run.

## 9.10 Omitted Proofs from Section 9.3

*Proof of Theorem 9.1.* We consider the case where payments are not allowed, i.e., $p_t(h, s) = 0$ for all $h \in \mathcal{H}$ and $s \in \mathcal{S}$. The case with payments and dynamic IR constraints is essentially the same. We use a similar reduction from MAX-SAT to the ones in [75, 81] for partially observable Markov decision processes (POMDPs). Given a MAX-SAT instance with $n$ variables $x_1, \ldots, x_n$ and $m$ clauses $c_1, \ldots, c_m$ where $c_i = \{\ell_{i,j}\}_{j \in [k_i]}$ and each $\ell_{i,j}$ is a literal, we construct a dynamic environment where $T = n$, $|\mathcal{S}| = m + 1$, and $|\mathcal{A}| = 2$. The goal is to show that the maximum utility is precisely the fraction of clauses that can be simultaneously satisfied. Without loss of generality, we assume that no clause contains both the positive literal and the negative literal of a same variable. We first describe $\mathcal{S}$ and $\mathcal{A}$. Each clause $c_i$ corresponds to a unique state in $\mathcal{S}$, $s_i$. In addition to these $m$ states, there is another state $s_0$. $\mathcal{A}$ consists of two actions: $a_{\text{pos}}$ and $a_{\text{neg}}$. The transition operator and the principal's valuation function are such that:

- The initial distribution is uniform over $\{s_i\}_{i \in [m]}$, i.e., $P_0(s_i) = 1/m$ for each $i \in [m]$.
- For each $t \in [T]$ and $a \in \mathcal{A}$,

$$P_t(s_0, a, s_0) = 1 \quad \text{and} \quad v_t^P(s_0, a) = 0.$$

  Moreover, for each $t \in [T]$ and $i \in [m]$:
  - If $x_t^+ \in c_i$, then

$$P_t(s_i, a_{\text{pos}}, s_0) = P_t(s_i, a_{\text{neg}}, s_i) = 1,$$

    and

$$v_t^P(s_i, a_{\text{pos}}) = 1 \quad \text{and} \quad v_t^P(s_i, a_{\text{neg}}) = 0.$$

- if $x_t^- \in c_i$, then
$$P_t(s_i, a_{\text{pos}}, s_i) = P_t(s_i, a_{\text{neg}}, s_0) = 1,$$
and
$$v_t^P(s_i, a_{\text{pos}}) = 0 \quad \text{and} \quad v_t^P(s_i, a_{\text{neg}}) = 1.$$
- otherwise,
$$P_t(s_i, a_{\text{pos}}, s_i) = P_t(s_i, a_{\text{neg}}, s_i) = 1,$$
and
$$v_t^P(s_i, a_{\text{pos}}) = v_t^P(s_i, a_{\text{neg}}) = 0.$$

- The principal and the agent are in a zero-sum situation, i.e., for any $t \in [T]$, $s \in \mathcal{S}$, $a \in \mathcal{A}$,
$$v_t^A(s, a) = 1 - v_t^P(s, a).$$

Now we show that the maximum utility is precisely the fraction of clauses that can be simultaneously satisfied. First observe that without loss of generality, an optimal mechanism depends only on time (and not on the reported states). This is because of the zero-sum situation: if the mechanism depends on the reports, then the agent can always choose the worst sequence of actions, which can only make the principal's utility smaller. Moreover, given the above observation, without loss of generality, an optimal mechanism is deterministic. This is because the overall utility of the principal is linear in the action at any time $t$, so one can always round a randomized mechanism into a deterministic one with at least the same overall utility.

Given the above observations, an optimal mechanism corresponds precisely to a way of assigning values to variables in the MAX-SAT instance: for each $t \in [T]$, the action at time $t$ is $a_{\text{pos}}$ iff the variable $x_t = 1$ (i.e., the literal $x_t^+$ is chosen). Moreover, when the initial state is $s_i$, the onward utility is $1$ if the clause $c_i$ is satisfied by the above assignment, and $0$ otherwise. Since the initial state is uniformly at random among $\{s_i\}_{i \in [m]}$, the maximum utility is precisely the maximum fraction of clauses that are satisfiable by some assignment. The theorem then follows from the fact that MAX-SAT is hard to approximate within a factor of $7/8 + \varepsilon$ for any $\varepsilon > 0$ [55]. □

*Proof of Lemma 9.1.* For brevity, let $\text{obj}$ denote the objective, i.e.,
$$\text{obj} = \sum_{h \in \mathcal{H}, s \in \mathcal{S}: (h,s) \text{ is feasible}} \left( \sum_{a \in \mathcal{A}} v_{|h|+1}^P(s, a) \cdot x(h, s, a) + y(h, s) \right).$$
Moreover, for each $h \in \mathcal{H}$, $s \in \mathcal{S}$, let
$$\text{obj}(h, s) = \sum_{h' \in \mathcal{H}, s' \in \mathcal{S}: (h,s) \sqsubseteq (h',s')} \left( \sum_{a \in \mathcal{A}} v_{|h'|+1}^P(s', a) \cdot x(h', s', a) + y(h', s') \right).$$
Observe that
$$\text{obj} = \sum_{s \in \mathcal{S}} \text{obj}(\emptyset, s).$$

We first prove inductively that for each $h \in \mathcal{H}$, $s \in \mathcal{S}$,

$$\mathrm{obj}(h, s) = z(h, s) \cdot u_P^M(h, s).$$

When $|h| = T - 1$, by the definition of feasible extensions and the construction of the mechanism,

$$\begin{aligned}
\mathrm{obj}(h, s) &= \sum_{a \in \mathcal{A}} v_T^P(s, a) \cdot x(h, s, a) + y(h, s) \\
&= z(h, s) \cdot \left( \sum_{a \in \mathcal{A}} v_T^P(s, a) \cdot \pi(h, s, a) + p(h, s) \right) \\
&= z(h, s) \cdot u_P^M(h, s).
\end{aligned}$$

When $|h| < T - 1$, for similar reasons,

$$\begin{aligned}
\mathrm{obj}(h, s) &= \sum_{h', s' : (h,s) \subseteq (h', s')} \left( \sum_{a \in \mathcal{A}} v_{|h'|+1}^P(s', a) \cdot x(h', s', a) + y(h', s') \right) \\
&= \sum_{a \in \mathcal{A}} v_{|h|+1}^P(s, a) \cdot x(h, s, a) + y(h, s) \\
&\quad + \sum_{h', s' : (h,s) \subseteq (h', s'), |h'| > |h|} \left( \sum_{a \in \mathcal{A}} v_{|h'|+1}^P(s', a) \cdot x(h', s', a) + y(h', s') \right) \\
&= z(h, s) \cdot \left( \sum_{a \in \mathcal{A}} v_{|h|+1}^P(s, a) \cdot \pi(h, s, a) + p(h, s) \right) \\
&\quad + \sum_{a', s'' : P_{|h|+1}(s, a', s'') > 0} \; \sum_{h', s' : (h + (s, a'), s'') \subseteq (h', s')} \left( \sum_{a \in \mathcal{A}} v_{|h'|+1}^P(s', a) \cdot x(h', s', a) + y(h', s') \right)
\end{aligned}$$

By the induction hypothesis, the second sum above is equal to

$$\begin{aligned}
&\sum_{a', s'' : P_{|h|+1}(s, a', s'') > 0} \mathrm{obj}(h + (s, a'), s'') \\
&= \sum_{a', s'' : P_{|h|+1}(s, a', s'') > 0} z(h + (s, a'), s'') \cdot u_P^M(h + (s, a'), s'') \\
&= \sum_{a', s'' : P_{|h|+1}(s, a', s'') > 0} x(h, s, a') \cdot P_{|h|+1}^E(s, a', s'') \cdot u_P^M(h + (s, a'), s'') \\
&= \sum_{a \in \mathcal{A}, s' \in \mathcal{S}} x(h, s, a) \cdot P_{|h|+1}(s, a, s') \cdot u_P^M(h + (s, a), s') \\
&= z(h, s) \cdot \sum_{a \in \mathcal{A}} \left( \pi(h, s, a) \cdot \sum_{s' \in \mathcal{S}} P_{|h|+1}(s, a, s') \cdot u_P^M(h + (s, a), s') \right).
\end{aligned}$$

Putting this back into the above expression for $\mathrm{obj}(h, s)$, we get

$$
\begin{aligned}
&\mathrm{obj}(h, s) \\
&= z(h, s) \left( \sum_{a \in \mathcal{A}} v^P_{|h|+1}(s, a) \cdot \pi(h, s, a) + p(h, s) \right) \\
&\quad + z(h, s) \sum_{a \in \mathcal{A}} \left( \pi(h, s, a) \cdot \sum_{s' \in \mathcal{S}} P_{|h|+1}(s, a, s') \cdot u^M_P(h + (s, a), s') \right) \\
&= z(h, s) \left( \sum_{a \in \mathcal{A}} \pi_{|h|+1}(h, s, a) \left( v^P_{|h|+1}(s, a) + \sum_{s' \in \mathcal{S}} P_{|h|+1}(s, a, s') \cdot u^M_P(h + (s, a), s') \right) + p(h, s) \right) \\
&= z(h, s) \cdot u^M_P(h, s).
\end{aligned}
$$

So for any $h \in \mathcal{H}$, $s \in \mathcal{S}$, $\mathrm{obj}(h, s) = z(h, s) \cdot u^M_P(h, s)$. Then we immediately have

$$
u^M_P(\emptyset) = \sum_{s \in \mathcal{S}} P_0(s) \cdot u^M_P(\emptyset, s) = \sum_{s \in \mathcal{S}} z(\emptyset, s) \cdot u^M_P(\emptyset, s) = \sum_{s \in \mathcal{S}} \mathrm{obj}(\emptyset, s) = \mathrm{obj}. \qquad \square
$$

*Proof of Lemma 9.3.* By Eq. (9.4) and Lemma 9.2, for all $h$, $s$, $s'$,

$$
\begin{aligned}
&u(h, s, s') \\
&= \sum_{a \in \mathcal{A}} v^A_{|h|+1}(s, a) \cdot x(h, s', a) - y(h, s') \\
&\quad + \sum_{a \in \mathcal{A}, s'' \in \mathcal{S}} \frac{P_{|h|+1}(s, a, s'')}{P^E_{|h|+1}(s', a, s'')} \cdot z(h + (s', a), s'') \cdot u^M_A(h + (s', a), s'') && \text{(Lemma 9.2)} \\
&= \sum_{a \in \mathcal{A}} v^A_{|h|+1}(s, a) \cdot x(h, s', a) - y(h, s') + \sum_{a \in \mathcal{A}, s'' \in \mathcal{S}} P_{|h|+1}(s, a, s'') \cdot x(h, s', a) \cdot u^M_A(h + (s', a), s'') \\
& && \text{(Eq. (9.4))}
\end{aligned}
$$

Now by rearranging the above expression and applying the construction of the mechanism $M$ and

the single-step reporting strategy $r_{h,s,s'}$, we have

$$
\begin{aligned}
& u(h,s,s') \\
&= \sum_{a\in\mathcal{A}} x(h,s',a)\left(v^A_{|h|+1}(s,a) + \sum_{s''\in\mathcal{S}} P_{|h|+1}(s,a,s'')\cdot u^M_A(h+(s',a),s'')\right) - y(h,s') \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(rearranging)} \\
&= z(h,s')\left(\sum_{a} \pi(h,s',a)\left(v^A_{|h|+1}(s,a) + \sum_{s''} P_{|h|+1}(s,a,s'')\cdot u^M_A(h+(s',a),s'')\right) - p(h,s')\right) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(construction of mechanism)} \\
&= z(h,s')\left(\sum_{a} \pi(h,s',a)\left(v^A_{|h|+1}(s,a) + \sum_{s''} P_{|h|+1}(s,a,s'')\cdot u_A^{M,r_{h,s,s'}}(h+(s',a),s'')\right) - p(h,s')\right) \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(construction of } r_{h,s,s'}\text{)} \\
&= z(h,s')\cdot u_A^{M,r_{h,s,s'}}(h,s), \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(definition of } u_A^{M,r_{h,s,s'}}\text{)}
\end{aligned}
$$

as desired. □

*Proof of Lemma 9.4.* Fix $h \in \mathcal{H}$, $s, s' \in \mathcal{S}$, and let $(s_p, a_p) = \text{last}(h)$. When $h = \emptyset$, by Lemmas 9.2 and 9.3 and Eq. (9.3),

$$
\begin{aligned}
& u(h,s) \geq \frac{P^E_{|h|}(s_p,a_p,s)}{P^E_{|h|}(s_p,a_p,s')}\cdot u(h,s,s') \\
\iff\ & z(\emptyset,s)\cdot u^M_A(\emptyset,s) \geq \frac{P^E_0(s_p,a_p,s)}{P^E_0(s_p,a_p,s')}\cdot z(\emptyset,s')\cdot u_A^{M,r_{\emptyset,s,s'}}(\emptyset,s) \\
\iff\ & z(\emptyset,s)\cdot u^M_A(\emptyset,s) \geq \frac{P^E_0(s)}{P^E_0(s')}\cdot z(\emptyset,s')\cdot u_A^{M,r_{\emptyset,s,s'}}(\emptyset,s) \\
\iff\ & u^M_A(\emptyset,s) \geq u_A^{M,r_{\emptyset,s,s'}}(\emptyset,s).
\end{aligned}
$$

When $|h| > 0$, suppose $h = (s_1, a_1, \ldots, s_t, a_t)$, and let $h_p = (s_1, a_1, \ldots, s_{t-1}, a_{t-1})$. By Lemmas 9.2 and 9.3 and Eq. (9.2),

$$
\begin{aligned}
& u(h,s) \geq \frac{P^E_{|h|}(s_p,a_p,s)}{P^E_{|h|}(s_p,a_p,s')}\cdot u(h,s,s') \\
\iff\ & z(h,s)\cdot u^M_A(h,s) \geq \frac{P^E_{|h|}(s_p,a_p,s)}{P^E_{|h|}(s_p,a_p,s')}\cdot z(h,s')\cdot u_A^{M,r_{h,s,s'}}(h,s) \\
\iff\ & x(h_p,s_p,a_p)\cdot u^M_A(h,s) \geq x(h_p,s_p,a_p)\cdot u_A^{M,r_{h,s,s'}}(h,s).
\end{aligned}
$$

Note that when $x(h_p, s_p, a_p) = 0$, $(h, s)$ cannot be reached, because (1) if $z(h_p, s_p) > 0$, then when the (reported) history-state pair is $(h_p, s_p)$, the mechanism never takes action $a_p$, and (2) if $z(h_p, s_p) = 0$, then such an impossible action exists somewhere in $h_p$. In such cases, $\pi(h, s)$ and

$p(h, s)$ will never be accessed, since it is impossible for the (reported) history to be $h$. In other words, when $(h, s)$ is reachable, we must have $x(h_p, s_p, a_p) > 0$, in which case the last inequality is equivalent to $u_A^M(h, s) \geq u_A^{M,r_{h,s,s'}}(h, s)$. □

*Proof of Lemma 9.5.* We only need to show that IC is equivalent to robustness against single-step misreporting. We prove this inductively, aiming to eliminate misreporting one step at a time. To be more specific, consider the following partial reporting strategy. For a reporting strategy $r$, $t \in [T]$, let $r|_{\geq t}$ denote the reporting strategy restricted to time $t, t+1, \ldots, T$, i.e., for any $h' \in \mathcal{H}$, $s' \in \mathcal{S}$,

$$r|_{\geq t}(h', s') = \begin{cases} s', \text{if } |h'| + 1 < t \\ r(h', s'), \text{otherwise} \end{cases}.$$

Similarly, let $r|_{<t}$ denote $r$ restricted to time $1, 2, \ldots, t-1$, and $r|_{=t}$ denote $r$ restricted to time $t$. We show inductively that for any reachable history-state pair $(h, s)$, and any reporting strategy $r$,

$$u_A^{M,(r|_{<|h|+1})}(h, s) \geq u_A^{M,r}(h, s).$$

Without loss of generality, we assume that for any unreachable pair $(h', s')$, $r$ simply reports truthfully, i.e., $r(h', s') = s'$.

Recall that $r(h)$ is the reported history given by $r$ when the true history is $h$. When $|h| = T-1$, the above claim is implied by Lemma 9.4, because

$$u_A^{M,r}(h, s) = u_A^{M,(r|_{\geq T})}(r(h), s) \geq u_A^M(r(h), s) = u_A^{M,(r|_{<T})}(h, s).$$

Now suppose $|h| < T - 1$. By the induction hypothesis, we have

$$u_A^{M,r}(h, s) = u_A^{M,(r|_{\geq |h|+1})}(r(h), s) \leq u_A^{M,((r|_{\geq |h|+1})|_{<|h|+2})}(r(h), s) = u_A^{M,(r|_{=|h|+1})}(r(h), s).$$

Now again by Lemma 9.4, we have

$$u_A^{M,r}(h, s) \leq u_A^{M,(r|_{=|h|+1})}(r(h), s) \leq u_A^M(r(h), s) = u_A^{M,(r|_{<|h|+1})}(h, s),$$

which establishes the above claim.

Now observe that as a special case of the claim, for any $s \in \mathcal{S}$,

$$u_A^{M,r}(\emptyset, s) \leq u_A^{M,(r|_{<1})}(\emptyset, s) = u_A^M(\emptyset, s).$$

Now summing over $s$, this implies that for any reporting strategy $r$,

$$u_A^{M,r}(\emptyset) = \sum_{s \in \mathcal{S}} P_0(s) \cdot u_A^{M,r}(\emptyset, s) \leq \sum_{s \in \mathcal{S}} P_0(s) \cdot u_A^M(\emptyset, s) = u_A^M(\emptyset).$$ □

*Proof of Theorem 9.2.* Given the correspondence between mechanisms and LP variables, by Lemma 9.5, it is easy to see that (modulo the unreachable parts) every IC and IR mechanism corresponds bijectively to a feasible solution to the LP in Figure 9.1. Moreover, by Lemma 9.1, the objective value of this solution is precisely the principal's overall utility, which directly implies that an optimal solution to the LP corresponds to an IC and IR mechanism which maximizes the principal's overall utility.

Now observe that the number of variables and the number of constraints in the LP are both $O(|\mathcal{S}|^{T+1}|\mathcal{A}|^T)$. Moreover, all relevant coefficients in the LP can be encoded using $O(L)$ bits. It is well-known that such an LP can be solved in time $\text{poly}(|\mathcal{S}|^T, |\mathcal{A}|^T, L)$. □

## 9.11 Omitted Proofs from Section 9.4

*Proof of Lemma 9.7.* We construct $M'$ explicitly based on $M$. Let $\pi'(t, s_p, a_p, s, a)$ be the probability that $M'$ chooses action $a$ at time $t$ in state $s$ when the previous state-action pair is $(s_p, a_p)$. Similarly, let $p'(t, s_p, a_p, s)$ be the payment specified by $M'$ at time $t$ in state $s$ when the previous state-action pair is $(s_p, a_p)$. We construct $M'$ from $M$ inductively as follows. For each $t \in [T]$, $s_p \in \mathcal{S}$ and $a_p \in \mathcal{A}$, let $h^*(t, s_p, a_p) \in \mathcal{H}_{t-1}$ be any history such that

$$h^*(t, s_p, a_p) \in \operatorname{argmax}_{h \in \mathcal{H}_{t-1}:(s_p,a_p)=\text{last}(h)} \sum_{s \in \mathcal{S}} P_{t-1}(s_p, a_p, s) \left( p(h, s) + \sum_{a \in \mathcal{A}} \pi(h, s, a) \left( v^P_{|h|+1}(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') u^M_P(h + (s, a), s') \right) \right).$$

Then, for all $s \in \mathcal{S}$, let

$$\pi'(t, s_p, a_p, s) = \pi(h^*(t, s_p, a_p), s) \quad \text{and} \quad p(t, s_p, a_p, s) = p(h^*(t, s_p, a_p), s).$$

This finishes the construction of $M'$.

We first show that $u^{M'}_P(\emptyset) \geq u^M_P(\emptyset)$, by inductively showing a stronger claim: for all $h \in \mathcal{H}$,

$$\sum_s P_{|h|}(s_p, a_p, s) \cdot u^{M'}_P(h, s) \geq \sum_s P_{|h|}(s_p, a_p, s) \cdot u^M_P(h, s),$$

where $(s_p, a_p) = \text{last}(h)$. For all $h \in \mathcal{H}_{T-1}$, letting $(s_p, a_p) = \text{last}(h)$, by the construction of $M'$, we have

$$\begin{aligned} \sum_s P_{T-1}(s_p, a_p, s) \cdot u^{M'}_P(h, s) &= \sum_s P_{T-1}(s_p, a_p, s) \cdot u^M_P(h^*(T, s_p, a_p), s) \\ &\geq \sum_s P_{T-1}(s_p, a_p, s) \cdot u^M_P(h, s). \end{aligned}$$

Now for all $h \in \mathcal{H}$ where $|h| < T - 1$, letting $(s_p, a_p) = \text{last}(h)$ and $h^* = h^*(|h| + 1, s_p, a_p)$, we

have

$$
\begin{aligned}
&\sum_s P_{|h|}(s_p, a_p, s) \cdot u_P^{M'}(h, s) \\
&= \sum_s P_{|h|}(s_p, a_p, s) \left( p(h^*, s) + \sum_{a \in \mathcal{A}} \pi(h^*, s, a) \left( v_{|h|+1}^P(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') \cdot u_P^{M'}(h + (s, a), s') \right) \right) \\
&= \sum_s P_{|h|}(s_p, a_p, s) \left( p(h^*, s) + \sum_{a \in \mathcal{A}} \pi(h^*, s, a) \left( v_{|h|+1}^P(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') \cdot u_P^{M'}(h^* + (s, a), s') \right) \right) \\
&\qquad \text{(property of } M') \\
&\geq \sum_s P_{|h|}(s_p, a_p, s) \left( p(h^*, s) + \sum_{a \in \mathcal{A}} \pi(h^*, s, a) \left( v_{|h|+1}^P(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') \cdot u_P^{M}(h^* + (s, a), s') \right) \right) \\
&\qquad \text{(induction hypothesis)} \\
&\geq \sum_s P_{|h|}(s_p, a_p, s) \left( p(h, s) + \sum_{a \in \mathcal{A}} \pi(h, s, a) \left( v_{|h|+1}^P(s, a) + \sum_{s' \in \mathcal{S}} P_t(s, a, s') \cdot u_P^{M}(h + (s, a), s') \right) \right) \\
&\qquad \text{(choice of } h^*) \\
&= \sum_s P_{|h|}(s_p, a_p, s) \cdot u_P^{M}(h, s).
\end{aligned}
$$

Then in particular, we have

$$u_P^{M'}(\emptyset) = \sum_s P_0(s) \cdot u_P^{M'}(\emptyset, s) \geq \sum_s P_0(s) \cdot u_P^{M}(\emptyset, s) = u_P^{M}(\emptyset).$$

Finally we prove that $M'$ is IC. By the proof of Lemma 9.5, we only need to show that $M'$ is robust against any single-step reporting strategy $r_{h,s,s'}$. In fact, letting $(s_p, a_p) = \text{last}(h)$ and $h^* = h^*(|h| + 1, s_p, a_p)$,

$$u_A^{M'}(h, s) = \sum_a \pi(h^*, s, a) \cdot v_{|h|+1}^A(s, a) + p(h^*, s) = u_A^{M}(h^*, s).$$

Moreover,

$$u_A^{M', r_{h,s,s'}}(h, s) = \sum_a \pi(h^*, s', a) \cdot v_{|h|+1}^A(s, a) + p(h^*, s) = u_A^{M, r_{h,s,s'}}(h^*, s).$$

Since $M$ is IC, we have

$$u_A^{M'}(h, s) = u_A^{M}(h^*, s) \geq u_A^{M, r_{h,s,s'}}(h^*, s) = u_A^{M', r_{h,s,s'}}(h, s).$$

Now by the argument in the proof of Lemma 9.5, we know that for all reporting strategy $r$, $h \in \mathcal{H}$, $s \in \mathcal{S}$,

$$u_A^{M', r}(h, s) \leq u_A^{M', (r|_{<|h|+1})}(h, s),$$

so

$$u_A^{M',(r|_{\geq|h|+1})}(h,s) \leq u_A^{M',((r|_{\geq|h|+1})|_{<|h|+1})}(h,s) = u_A^{M'}(h,s),$$

which is precisely the IC requirement for myopic agents. Similar arguments guarantee that $M'$ has the same IR property as $M$. □

*Proof of Theorem 9.4.* We first argue the easy part, i.e., the time complexity. Observe that calls to OptStatMech dominates the time complexity. Moreover, the algorithm makes $T|\mathcal{S}||\mathcal{A}|$ calls to OptStatMech, so the overall time complexity is as stated.

Now we show the optimality of the computed mechanism $M$. We prove inductively a stronger claim, i.e., for any $t \in [T]$, $s_p \in \mathcal{S}$, $a_p \in \mathcal{A}$,

$$\sum_s P_0(s_p, a_p, s) \cdot u_P^M(t, s_p, a_p, s) = \max_{M'} P_0(s_p, a_p, s) \cdot u_P^{M'}(t, s_p, a_p, s),$$

where the maximum is over all succinct mechanisms $M'$ that are IC and (optionally) IR. First observe that for all $s \in \mathcal{S}$, $a \in \mathcal{A}$,

$$u(T, s, a) = v_T^P(s, a).$$

So, for all $s_p \in \mathcal{S}$, $a_p \in \mathcal{A}$,

$$\begin{aligned}
&\sum_s P_0(s_p, a_p, s) \cdot u_P^M(T, s_p, a_p, s) \\
&= \sum_s P_0(s_p, a_p, s) \cdot \left( p(T, s_p, a_p, s) + \sum_a \pi(T, s_p, a_p, s, a) \cdot v_T^P(s, a) \right) \\
&= \max_{M'=(\pi', p')} \sum_s P_0(s_p, a_p, s) \cdot \left( p'(T, s_p, a_p, s) + \sum_a \pi'(T, s_p, a_p, s, a) \cdot v_T^P(s, a) \right) \\
&\qquad\qquad \text{(optimality of } M \text{ at time } T \text{ as a static mechanism)} \\
&= \max_{M'} \sum_s P_0(s_p, a_p, s) \cdot u_P^M(T, s_p, a_p, s).
\end{aligned}$$

Again, the maximum is over all succinct mechanisms $M'$ that are IC and (optionally) IR.

Now for $t \in [T-1]$, by the construction of $M$,

$$\begin{aligned}
&\sum_s P_0(s_p, a_p, s) \cdot u_P^M(t, s_p, a_p, s) \\
&= \sum_s P_0(s_p, a_p, s) \cdot \left( p(t, s_p, a_p, s) + \sum_a \pi(t, s_p, a_p, s, a) \cdot \left( v_t^P(s, a) \right.\right. \\
&\quad \left.\left. + \sum_{s'} P_t(s, a, s') \cdot u_P^M(t+1, s, a, s') \right) \right) \\
&= \max_{M'=(\pi', p')} \sum_s P_0(s_p, a_p, s) \cdot \left( p'(t, s_p, a_p, s) + \sum_a \pi'(t, s_p, a_p, s, a) \cdot \left( v_t^P(s, a) \right.\right. \\
&\quad \left.\left. + \sum_{s'} P_t(s, a, s') \cdot u_P^M(t+1, s, a, s') \right) \right). \quad \text{(optimality of } M \text{ at time } t \text{ as a static mechanism)}
\end{aligned}$$

By the induction hypothesis and the fact that $M'$ is succinct,

$$\begin{aligned}
&\sum_s P_0(s_p, a_p, s) \cdot u_P^M(t, s_p, a_p, s) \\
&= \max_{M'=(\pi', p')} \sum_s P_0(s_p, a_p, s) \cdot \left( p'(t, s_p, a_p, s) + \sum_a \pi'(t, s_p, a_p, s, a) \cdot \left( v_t^P(s, a) \right.\right. \\
&\quad \left.\left. + \max_{M''} \sum_{s'} P_t(s, a, s') \cdot u_P^{M''}(t+1, s, a, s') \right) \right) \qquad \text{(induction hypothesis)} \\
&= \max_{M'=(\pi', p')} \sum_s P_0(s_p, a_p, s) \cdot \left( p'(t, s_p, a_p, s) + \sum_a \pi'(t, s_p, a_p, s, a) \cdot \left( v_t^P(s, a) \right.\right. \\
&\quad \left.\left. + \sum_{s'} P_t(s, a, s') \cdot u_P^{M'}(t+1, s, a, s') \right) \right) \qquad (M' \text{ is succinct}) \\
&= \max_{M'} \sum_s P_0(s_p, a_p, s) \cdot u_P^{M'}(t, s_p, a_p, s).
\end{aligned}$$

All maxima are over all succinct mechanisms that are IC and (optionally) IR. As a result, we have

$$u_P^M(\emptyset) = \sum_s P_0(s) \cdot u_P^M(\emptyset, s) = \max_{M'} \sum_s P_0(s) \cdot u_P^{M'}(\emptyset, s) = \max_{M'} u_P^{M'}(\emptyset). \qquad \square$$

## 9.12 Omitted Proofs from Section 9.8

*Proof of Theorem 9.5.* First suppose the agent is patient and without loss of generality has a discount factor of 1. Let $T = 2$ and $\mathcal{S} = \mathcal{A} = [n]$ where $n \geq \varepsilon^{-1}$. The initial distribution is uniform over $[n]$, i.e., $P_0(i) = 1/n$ for all $i \in [n]$, i.e., no matter what action is played, all states always

transition to state 1. The transition operator is such that $P_1(i, j, 1) = 1$ for all $i, j \in [n]$. At time $T = 2$, the principal's valuations are $v_T^P(i, j) = 0$ for all $i, j \in [n]$. At time 1, the principal's valuation function is such that for all $i, j \in [n]$, $v_1^P(i, j) = 1$ if $i = j$, and $v_1^P(i, j) = 0$ if $i \neq j$. For $t \in [T]$, the agent's valuation function is such that for all $i, j \in [n]$, $v_t^A(i, j) = 0$ if $i = j$, and $v_t^A(i, j) = 1$ if $i \neq j$.

Consider the principal's optimal utility, which is clearly upper bounded by 1 (1 at time 1 and 0 at time 2). The following mechanism is IC and achieves this upper bound:

- At time 1, play action $i$ for each state $i \in [n]$.
- At time $T = 2$, play action $(i \bmod n) + 1$ iff the state at time 1 is $i$.

The mechanism is IC because regardless of the (reported) initial state, the agent achieves overall utility 1. It is easy to check this mechanism achieves utility 1.

On the other hand, any memoryless IC mechanism can achieve utility at most $1/n \leq \varepsilon$. This is because at time $T = 2$, the current state provides absolutely no information, so the mechanism has to perform the same (randomized) action regardless of the initial state. As a result, in order to be IC, the mechanism has to satisfy the following condition at time 1: for all $i, j \in [n]$, $\pi(i, i) \leq \pi(j, i)$, where $\pi(a, b)$ is the probability that action $b$ is played in state $a$ at time 1. So the principal's utility can be bounded as follows:

$$\frac{1}{n} \sum_i \pi(i, i) \leq \frac{1}{n} \sum_i \left( \frac{1}{n} \sum_j \pi(j, i) \right) = \frac{1}{n^2} \sum_{i,j} \pi(j, i) = \frac{1}{n}.$$

This concludes the proof when the agent is patient.

Now consider the case with a myopic agent. Again, let $T = 2$ and $\mathcal{S} = \mathcal{A} = [n]$ where $n \geq \varepsilon^{-1}$. The initial distribution is again uniform over $[n]$, i.e., $P_0(i) = 1/n$ for all $i \in [n]$. The transition operator is such that $P_1(i, j, i) = 1$ for all $i, j \in [n]$, i.e., no matter what action is played, state $i$ always transitions to state $i$. At time 1, the principal's and the agent's valuations are $v_1^P(i, j) = v_1^A(i, j) = 0$ for all $i, j \in [n]$. At time $T = 2$, the principal's valuation function is such that for all $i, j \in [n]$, $v_T^P(i, j) = 1$ if $i = j$, and $v_T^P(i, j) = 0$ if $i \neq j$. And the agent's valuation function is such that for all $i, j \in [n]$, $v_T^A(i, j) = 0$ if $i = j$, and $v_T^A(i, j) = 1$ if $i \neq j$.

The principal's optimal utility, 1, is achieved by the following succinct (but not memoryless) IC mechanism:

- At time 1, play action 1 for all states.
- At time 2, play action $i$ iff the state at time 1 is $i$.

The mechanism is IC in particular because the agent is myopic and cannot change the past. It is easy to check this mechanism achieves utility 1.

On the other hand, any memoryless IC mechanism can achieve utility at most $1/n \leq \varepsilon$. This is because at time $T = 2$, the mechanism cannot memorize anything before, so it has to be IC based only on the current state, which puts the mechanism in a situation that is essentially the same as at time 1 in the hard instance for patient agents. Similar arguments then guarantee that the principal's utility is at most $1/n$, which concludes the proof for myopic agents. Finally, we note that the above constructions work even if payments are allowed. □

*Proof of Theorem 9.6.* We use reductions from MAX-SAT similar to that in Theorem 9.1 for both myopic and patient agents. First consider the case where the agent is patient with a discount factor of 1. In this case, the reduction in Theorem 9.1 applies without any modification. In particular, since the principal and the agent are in a zero-sum situation, without loss of generality, any optimal memoryless mechanism does not depend on the reported states. And again, since the principal's utility is multilinear in the actions at each time, there is a deterministic mechanism which is optimal. As argued in the proof of Theorem 9.1, such a mechanism corresponds precisely to an optimal assignment of variables in the MAX-SAT instance, which implies the $7/8 + \varepsilon$ inapproximability.

Now consider the case where the agent is myopic. Here we slightly modify the reduction, and in particular, the agent's valuation functions. That is, for each $t \in [T]$ and $i \in [m]$, we let

$$v_t^A(s, a_{\text{pos}}) = c \quad \text{and} \quad v_t^A(s, a_{\text{neg}}) = 0,$$

for all $s \in \mathcal{S}$, where $c > 0$ is an arbitrarily small constant. This guarantees that at any time $t$, in order to be IC, the (randomized) actions for all states have to be exactly the same. Then since the principal's utility is multilinear, again it is without loss of generality to consider deterministic mechanisms, which correspond to assignments of variables. The ratio of $7/8 + \varepsilon$ follows immediately. Finally, we remark that the above reductions still work when payments are allowed. □

**objective:** $$\max \sum_{h \in \mathcal{H}, s \in \mathcal{S}: (h,s) \text{ is feasible}} \left( \sum_{a \in \mathcal{A}} v^P_{|h|+1}(s,a) \cdot x(h,s,a) + y(h,s) \right) \tag{9.1}$$

**flow constraints:**
$$z(h,s) = \sum_{a \in \mathcal{A}} x(h,s,a) \qquad \forall h \in \mathcal{H}, s \in \mathcal{S} \tag{9.2}$$
$$z(\emptyset, s) = P^E_0(s) \qquad \forall s \in \mathcal{S} \tag{9.3}$$
$$z(h + (s,a), s') = P^E_{|h|+1}(s,a,s') \cdot x(h,s,a) \qquad \forall h \in \mathcal{H}, s, s' \in \mathcal{S}, a \in \mathcal{A} \tag{9.4}$$

**utility:**
$$u(h,s) = \sum_{h' \in \mathcal{H}, s' \in \mathcal{S}: (h,s) \subseteq (h',s')} \left( \sum_{a \in \mathcal{A}} v^A_{|h'|+1}(s',a) \cdot x(h',s',a) - y(h',s') \right) \quad \forall h \in \mathcal{H}, s \in \mathcal{S} \tag{9.5}$$

**IC constraints:**
$$u(h,s,s') = \sum_{a \in \mathcal{A}} v^A_{|h|+1}(s,a) \cdot x(h,s',a) - y(h,s') + \sum_{a \in \mathcal{A}, s'' \in \mathcal{S}} \frac{P_{|h|+1}(s,a,s'')}{P^E_{|h|+1}(s',a,s'')} \cdot u(h + (s',a), s'') \qquad \forall h \in \mathcal{H}, s, s' \in \mathcal{S} \tag{9.6}$$

$$u(h,s) \geq \frac{P^E_{|h|}(s_p, a_p, s)}{P^E_{|h|}(s_p, a_p, s')} \cdot u(h,s,s'), \text{where } (s_p, a_p) = \text{last}(h) \quad \forall h \in \mathcal{H}, s, s' \in \mathcal{S} \tag{9.7}$$

**IR constraints:**
$$u(h,s) \geq 0 \qquad \forall h \in \mathcal{H}, s \in \mathcal{S} \tag{9.8}$$

**feasible actions:**
$$x(h,s,a) \geq 0 \qquad \forall h \in \mathcal{H}, s \in \mathcal{S}, a \in \mathcal{A} \tag{9.9}$$

**feasible payments:**
$$y(h,s) \geq 0 \qquad \forall h \in \mathcal{H}, s \in \mathcal{S} \tag{9.10}$$

Figure 9.1: Linear program for computing an optimal dynamic mechanism.

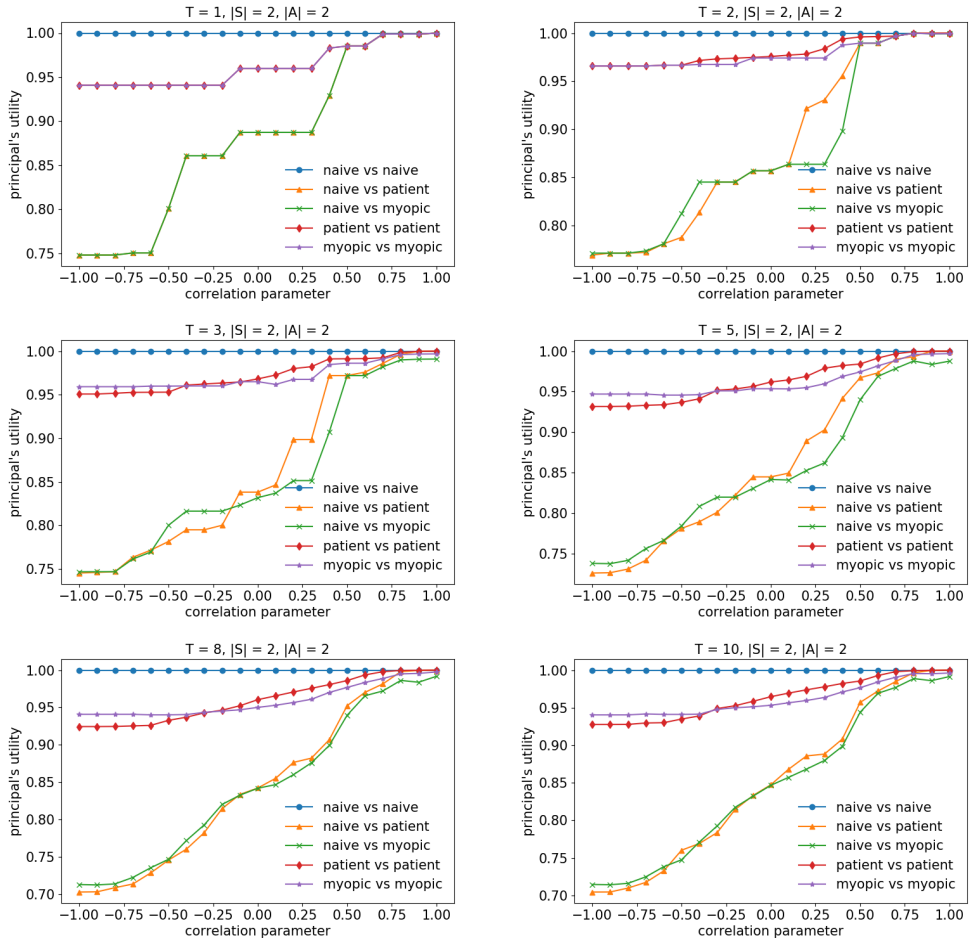


Figure 9.2: Performance of different mechanisms facing different types of agents when $|\mathcal{S}| = |\mathcal{A}| = 2$ and the time horizon $T$ varies. All numbers are normalized by the optimal utility facing a naïve agent. Every point is an average of 10 independent runs using different random seeds.

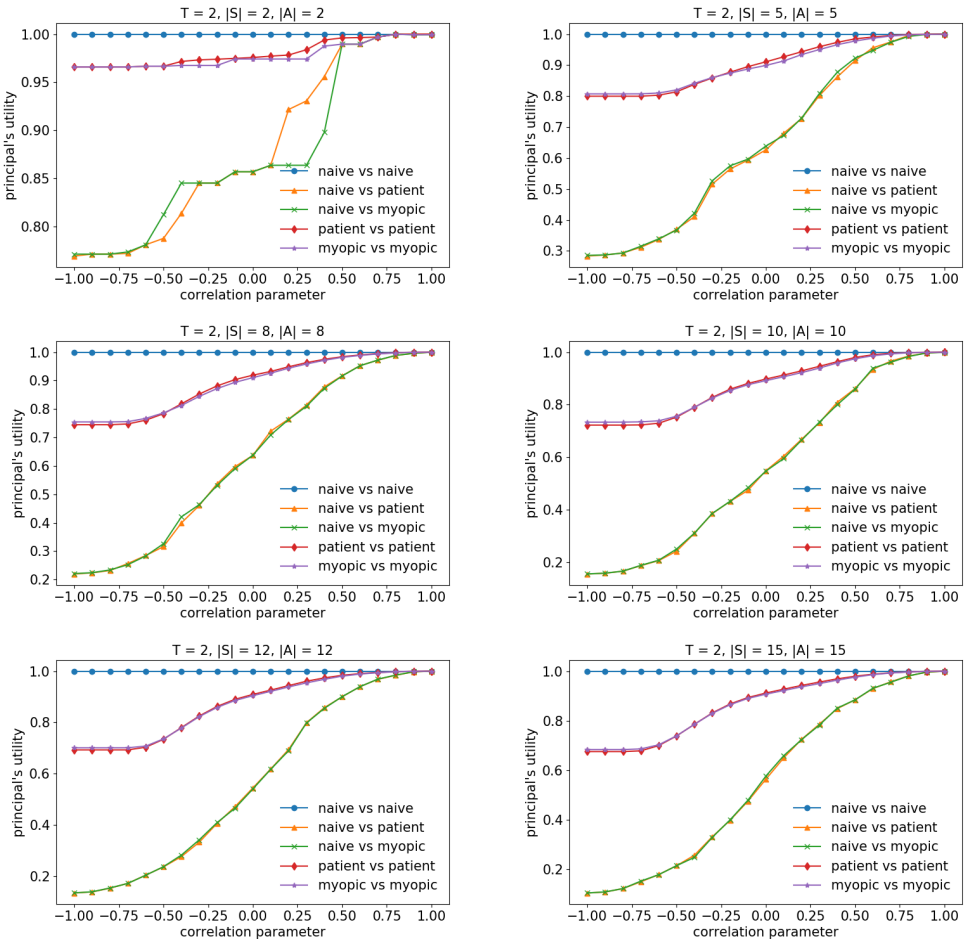


Figure 9.3: Performance of different mechanisms facing different types of agents when $T = 2$ and the numbers of states and actions, $|\mathcal{S}|$ and $|\mathcal{A}|$, vary. All numbers are normalized by the optimal utility facing a naïve agent. Every point is an average of 10 independent runs using different random seeds.

# Chapter 10

# Conclusion and Future Directions

In this dissertation, I presented structural, computational, and experimental results for a number of key problems in strategic machine learning. Conceptually, these results show the possibility of, and provide high-level guiding pinciples for, designing efficient and accurate machine learning algorithms in the presence of strategic behavior. Technically, these results combine techniques in machine learning and mechanism design, which highlights the importance of interdisciplinary expertise in the design and analysis of machine learning algorithms in strategic environments. The principles and techniques presented in the dissertation will likely be useful in other problems in strategic machine learning, as well as real-world applications. Below we also discuss a few important future directions.

**Generic and modular tools for incentive-aware machine learning.** One of the most pressing issues in real-world machine learning in the presence of strategic behavior is the following: How to make machine learning algorithms already in use — many of which do not take users' incentives into consideration — robust against strategic manipulation, while keeping the cost of migration acceptable? While we do have satisfactory solutions for various applications already, deploying these solutions would often involve abandoning the existing infrastructure and rebuilding the entire system from scratch, which we would like to avoid in practice. An alternative, more realistic approach is to design add-on solutions that work in a generic and modular way, and then implement them on top of existing ones. For example, such add-on solutions could be procedures that preprocess the training data, so the existing model, when trained on the preprocessed data, becomes more robust against strategic manipulation; or, they could be procedures that postprocess the output of the existing model, so the combined pipeline becomes more robust. Designing such solutions will likely require deep understanding of both how existing systems work, and key techniques for achieving robustness against strategic manipulation.

**Reinforcement learning in the presence of strategic behavior.** In Section 1.3, Chapter 8 and Chapter 9, I have discussed how to make optimal decisions in dynamic environments in the presence of strategic behavior. It is then natural to ask the following question: What if the decision maker does not even have enough information about the environment itself, but instead has to

explore it by taking actions? For example, consider ride-sharing again: In order to optimally assign tasks, the ride-sharing platform needs to know the frequency of tasks between any pair of locations, as well as the traffic conditions at any time of the day. It is unrealistic to assume the platform has all this information, especially when it first starts off. So, the platform has to explore the environment as it operates, and gradually improve its decisions as more and more information becomes available. Importantly, the process of exploration is also affected by strategic behavior. Such problems are traditionally studied in the area of *reinforcement learning*. However, most (if not all) known approaches fail in the presence of strategic behavior. To design good algorithms for reinforcement learning in the presence of strategic behavior, one needs to combine techniques in reinforcement learning with insights from dynamic mechanism design.

# Bibliography


[1] Caralee J. Adams. *In Race for Test-Takers, ACT Outscores SAT—for Now*, 2017. URL `https://www.edweek.org/ew/articles/2017/05/24/in-race-for-test-takers-act-outscores-sat--for.html`. 4.1

[2] AM Andrew. Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters*, 9(5):216–219, 1979. 3.6

[3] Itai Ashlagi, Constantinos Daskalakis, and Nima Haghpanah. Sequential mechanisms with ex-post participation guarantees. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 213–214, 2016. 9.1.2

[4] Susan Athey and Ilya Segal. An efficient dynamic mechanism. *Econometrica*, 81(6):2463–2485, 2013. 9.1.2

[5] Vincenzo Auletta, Paolo Penna, Giuseppe Persiano, and Carmine Ventre. Alternatives to truthfulness are hard to recognize. *Autonomous Agents and Multi-Agent Systems*, 22(1): 200–216, 2011. 3.1, 3.1.1

[6] Pranjal Awasthi, Avrim Blum, Nika Haghtalab, and Yishay Mansour. Efficient PAC Learning from the Crowd. In *Conference on Learning Theory*, pages 127–150, 2017. 6.6

[7] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. Sample complexity of automated mechanism design. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2091–2099, 2016. 9.1.2

[8] Maria-Florina Balcan, Tuomas Sandholm, and Ellen Vitercik. A general theory of sample complexity for multi-item profit maximization. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 173–174, 2018. 9.1.2

[9] David P Baron and David Besanko. Regulation and information in a continuing relationship. *Information Economics and policy*, 1(3):267–302, 1984. 9.1.2

[10] Tugkan Batu, Lance Fortnow, Ronitt Rubinfeld, Warren D Smith, and Patrick White. Testing that distributions are close. In *Proceedings of the 41st IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 259–269, 2000. 5.4.1

[11] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, 6(5):679–684, 1957. 9.1

[12] Dirk Bergemann and Juuso Välimäki. The dynamic pivot mechanism. *Econometrica*, 78 (2):771–789, 2010. 9.1.2

[13] Dirk Bergemann and Juuso Välimäki. Dynamic mechanism design: An introduction. *Journal of Economic Literature*, 57(2):235–74, 2019. 9.1.2

[14] Avrim Blum, Nika Haghtalab, Ariel D Procaccia, and Mingda Qiao. Collaborative PAC Learning. In *Advances in Neural Information Processing Systems*, pages 2392–2401, 2017. 6.6

[15] Sébastien Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015. 1

[16] Andrea Celli, Stefano Coniglio, and Nicola Gatti. Private bayesian persuasion with sequential games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1886–1893, 2020. 9.1.2

[17] Siu-On Chan, Ilias Diakonikolas, Paul Valiant, and Gregory Valiant. Optimal algorithms for testing closeness of discrete distributions. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1193–1203, 2014. 5.4.1

[18] Nitesh V Chawla, Nathalie Japkowicz, and Aleksander Kotcz. Special issue on learning from imbalanced data sets. *ACM SIGKDD explorations newsletter*, 6(1):1–6, 2004. 4.6

[19] Wei Chen, Shang-Hua Teng, and Hanrui Zhang. Capturing Complementarity in Set Functions by Going Beyond Submodularity/Subadditivity. In *Innovations in Theoretical Computer Science (ITCS)*, 2019. 1.4

[20] Yiling Chen, Nicole Immorlica, Brendan Lucier, Vasilis Syrgkanis, and Juba Ziani. Optimal data acquisition for statistical estimation. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 27–44. ACM, 2018. 6.6

[21] Yiling Chen, Chara Podimata, Ariel D Procaccia, and Nisarg Shah. Strategyproof linear regression in high dimensions. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 9–26. ACM, 2018. 6.6

[22] Jeremy M. Cohen, Elan Rosenfeld, and J. Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, 2019. 1

[23] Michael B Cohen, Yin Tat Lee, and Zhao Song. Solving linear programs in the current matrix multiplication time. In *Proceedings of the 51st annual ACM SIGACT symposium on theory of computing*, pages 938–942, 2019. 8

[24] Vincent Conitzer and Tuomas Sandholm. Complexity of mechanism design. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 103–110, 2002. 3.1, 3.1.1, 3.2.2, 9.1, 9.1.2, 9.4.2, 9.6, 12

[25] Vincent Conitzer and Tuomas Sandholm. Self-interested automated mechanism design and implications for optimal combinatorial auctions. In *Proceedings of the 5th ACM conference on Electronic commerce*, pages 132–141, 2004. 3.1, 3.1.1, 9.1, 9.1.2, 9.6

[26] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, pages 82–90, 2006. 9.1, 9.4.2, 12

[27] Vincent Conitzer, Debmalya Panigrahi, and Hanrui Zhang. Learning Opinions in Social

Networks. In *International Conference on Machine Learning (ICML)*, 2020. 1.4

[28] Vincent Conitzer, Debmalya Panigrahi, and Hanrui Zhang. Learning Influence Adoption in Heterogeneous Networks. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 1.4

[29] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009. 4.1

[30] Pascal Courty and Li Hao. Sequential screening. *The Review of Economic Studies*, 67(4): 697–717, 2000. 9.1, 9.1.2

[31] Daniel Cullina, Arjun Nitin Bhagoji, and Prateek Mittal. Pac-learning in the presence of adversaries. In *Advances in Neural Information Processing Systems*, pages 230–241, 2018. 5

[32] Yuan Deng and Hanrui Zhang. Prior-independent Dynamic Auctions for a Value-maximizing Buyer. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 1.4

[33] Yuan Deng, Debmalya Panigrahi, and Hanrui Zhang. Online Combinatorial Auctions. In *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2021. 1.4

[34] Yuan Deng, Vahab Mirrokni, and Hanrui Zhang. Posted Pricing and Dynamic Prior-independent Mechanisms with Value Maximizers. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1.4

[35] Yuan Deng, Jieming Mao, Vahab S. Mirrokni, Hanrui Zhang, and Song Zuo. Autobidding auctions in the presence of user costs. In *ACM Web Conference (WWW)*, 2023. 1.4

[36] Jinshuo Dong, Aaron Roth, Zachary Schutzman, Bo Waggoner, and Zhiwei Steven Wu. Strategic classification from revealed preferences. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 55–70. ACM, 2018. 6.6

[37] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. `http://archive.ics.uci.edu/ml`, last accessed 07/09/2020. 4.6

[38] P Dütting, F Fischer, P Jirapinyo, J Lai, B Lubin, and DC Parkes. Payment rules through discriminant-based classifiers. *ACM Transactions on Economics and Computation*, 2015. 9.1.2

[39] Paul Dütting, Zhe Feng, Harikrishna Narasimhan, David Parkes, and Sai Srivatsa Ravindranath. Optimal auctions through deep learning. In *International Conference on Machine Learning*, pages 1706–1715. PMLR, 2019. 9.1.2

[40] Jeffrey C Ely. Beeps. *American Economic Review*, 107(1):31–53, 2017. 9.1.2

[41] Péter Eső and Balazs Szentes. Optimal information disclosure in auctions and the handicap auction. *The Review of Economic Studies*, 74(3):705–731, 2007. 9.1.2

[42] Usama M. Fayyad and Keki B. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In Ruzena Bajcsy, editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence. Chambéry, France, August 28 - September 3, 1993*, pages 1022–1029. Morgan Kaufmann, 1993. URL `http://ijcai.`

org/Proceedings/93-2/Papers/022.pdf. 4.6

[43] Zhe Feng, Harikrishna Narasimhan, and David C Parkes. Deep learning for revenue-optimal auctions with budgets. In *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pages 354–362, 2018. 9.1.2

[44] Raquel Florez-Lopez. Effects of missing data in credit risk scoring. a comparative analysis of methods to achieve robustness in the absence of sufficient data. *The Journal of the Operational Research Society*, 61(3):486–501, 2010. 4.1

[45] Gary L Friedman. The trustworthy digital camera: Restoring credibility to the photographic image. *IEEE Transactions on consumer electronics*, 39(4):905–910, 1993. 5.1

[46] Jiarui Gan, Rupak Majumdar, Goran Radanovic, and Adish Singla. Bayesian persuasion in sequential decision-making. *arXiv preprint arXiv:2106.05137*, 2021. 9.1.2

[47] Ganesh Ghalme, Vineet Nair, Itay Eilat, Inbal Talgam-Cohen, and Nir Rosenfeld. Strategic Classification in the Dark. In *International Conference on Machine Learning (ICML)*, 2021. 1.1

[48] Gagan Goel, Renato Paes Leme, Jon Schneider, David R. M. Thompson, and Hanrui Zhang. Eligibility mechanisms: Auctions meet information retrieval. In *ACM Web Conference (WWW)*, 2023. 1.4

[49] Shafi Goldwasser, Silvio Micali, and Ronald L Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal on Computing*, 17(2):281–308, 1988. 5.1

[50] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *CoRR*, abs/1412.6572, 2014. 1

[51] Jerry R Green and Jean-Jacques Laffont. Partially verifiable information and mechanism design. *The Review of Economic Studies*, 53(3):447–456, 1986. 3.1, 3.2.1

[52] Martin Grötschel, László Lovász, and Alexander Schrijver. Geometric methods in combinatorial optimization. In *Progress in combinatorial optimization*, pages 167–183. Elsevier, 1984. 3.4.3

[53] Mohammad Taghi Hajiaghayi, Robert Kleinberg, and Tuomas Sandholm. Automated online mechanism design and prophet inequalities. In *AAAI*, volume 7, pages 58–65, 2007. 9.1.2

[54] Sergiu Hart and Philip J Reny. Maximal revenue with multiple goods: Nonmonotonicity and other observations. *Theoretical Economics*, 10(3):893–922, 2015. 5.3.1

[55] Johan Håstad. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4): 798–859, 2001. 9.10

[56] Ronald A Howard. Dynamic programming and markov processes. 1960. 9.1

[57] Lily Hu, Nicole Immorlica, and Jennifer Wortman Vaughan. The disparate effects of strategic manipulation. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 259–268. ACM, 2019. 6.6

[58] Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran,

and Aleksander Madry. Adversarial examples are not bugs, they are features. *ArXiv*, abs/1905.02175, 2019. 1

[59] Shahin Jabbari, Ryan M Rogers, Aaron Roth, and Steven Z Wu. Learning from rational behavior: Predicting solutions to unknown linear programs. In *Advances in Neural Information Processing Systems*, pages 1570–1578, 2016. 6.6

[60] Steven Jecmen, Hanrui Zhang, Ryan Liu, Nihar B. Shah, Vincent Conitzer, and Fei Fang. Mitigating Manipulation in Peer Review via Randomized Reviewer Assignments. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020. 1.4

[61] Steven Jecmen, Hanrui Zhang, Ryan Liu, Fei Fang, Vincent Conitzer, and Nihar B. Shah. Near-Optimal Reviewer Splitting in Two-Phase Paper Reviewing and Conference Experiment Design. In *AAAI Conference on Human Computation and Crowdsourcing (HCOMP)*, 2022. 1.4

[62] Jongbin Jung, Connor Concannon, Ravi Shroff, Sharad Goel, and Daniel G Goldstein. Simple rules for complex decisions. *Available at SSRN 2919024*, 2017. 4.4

[63] Debarun Kar, Thanh H Nguyen, Fei Fang, Matthew Brown, Arunesh Sinha, Milind Tambe, and Albert Xin Jiang. Trends and applications in Stackelberg security games. *Handbook of dynamic game theory*, pages 1–47, 2017. 8.1.1

[64] Andrew Kephart and Vincent Conitzer. Complexity of mechanism design with signaling costs. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, pages 357–365, 2015. 3.1, 3.1.1, 9.1.2

[65] Andrew Kephart and Vincent Conitzer. The revelation principle for mechanism design with reporting costs. In *Proceedings of the 2016 ACM Conference on Economics and Computation*, pages 85–102, 2016. 3.1, 3.1.1, 3.2.1, 9.1.2, 1

[66] Anilesh K Krishnaswamy, Haoming Li, David Rein, Hanrui Zhang, and Vincent Conitzer. Classification with Strategically Withheld Data. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. (document), 1, 1.1, 1.4, 7, 4.3, 4.4, 4.6, 4.6, 11, 4.6, 4.6

[67] Tosca Lechner and Ruth Urner. Learning Losses for Strategic Classification. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. 1.1

[68] Yin Tat Lee and Aaron Sidford. Path finding methods for linear programming: Solving linear programs in $\widetilde{O}(\sqrt{\mathrm{rank}})$ iterations and faster algorithms for maximum flow. In *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, pages 424–433. IEEE, 2014. 8

[69] Stefan Lessmann, Bart Baesens, Hsin-Vonn Seow, and Lyn C Thomas. Benchmarking state-of-the-art classification algorithms for credit scoring: An update of research. *European Journal of Operational Research*, 247(1):124–136, 2015. 4.6, (a)

[70] Sagi Levanon and Nir Rosenfeld. Strategic Classification Made Practical. In *International Conference on Machine Learning (ICML)*, 2021. 1.1

[71] Sagi Levanon and Nir Rosenfeld. Generalized Strategic Classification and the Case of Aligned Incentives. In *International Conference on Machine Learning (ICML)*, 2022. 1.1

[72] Annie Liang, Xiaosheng Mu, and Vasilis Syrgkanis. Optimal and myopic information acquisition. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 45–46. ACM, 2018. 6.6

[73] R. Meir, Ariel D. Procaccia, and Jeffrey S. Rosenschein. Algorithms for strategyproof classification. *Artif. Intell.*, 186:123–156, 2012. 1.1

[74] Vahab Mirrokni, Renato Paes Leme, Pingzhong Tang, and Song Zuo. Non-clairvoyant dynamic mechanism design. *Econometrica*, 88(5):1939–1963, 2020. 9.1.2

[75] Martin Mundhenk, Judy Goldsmith, Christopher Lusena, and Eric Allender. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM (JACM)*, 47(4): 681–720, 2000. 8.1.2, 8.2.2, 9.10

[76] Roger B Myerson. Optimal auction design. *Mathematics of operations research*, 6(1): 58–73, 1981. 9.1, 9.1.2

[77] Roger B Myerson. Multistage games with communication. *Econometrica: Journal of the Econometric Society*, pages 323–358, 1986. 9.2

[78] Vineet Nair, Ganesh Ghalme, Inbal Talgam-Cohen, and Nir Rosenfeld. Strategic Representation. In *International Conference on Machine Learning (ICML)*, 2022. 1.1

[79] Harikrishna Narasimhan, Shivani Brinda Agarwal, and David C Parkes. Automated mechanism design without money via machine learning. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 2016. 9.1.2

[80] Christos Papadimitriou, George Pierrakos, Christos-Alexandros Psomas, and Aviad Rubinstein. On the complexity of dynamic mechanism design. In *Proceedings of the twenty-seventh annual ACM-SIAM symposium on Discrete algorithms*, pages 1458–1475. SIAM, 2016. 8.4, 9.1.2

[81] Christos H Papadimitriou and John N Tsitsiklis. The complexity of Markov decision processes. *Mathematics of operations research*, 12(3):441–450, 1987. 8.1.2, 8.2.2, 9.10

[82] Alessandro Pavan. Dynamic mechanism design: Robustness and endogenous types. In *Advances in Economics and Econometrics: Eleventh World Congress*, pages 1–62, 2017. 9.1.2

[83] Alessandro Pavan, Ilya Segal, and Juuso Toikka. Dynamic mechanism design: A myersonian approach. *Econometrica*, 82(2):601–653, 2014. 9.1, 9.1.2

[84] Karl Pearson. Contributions to the mathematical theory of evolution. ii. skew variation in homogeneous material. *Philosophical Transactions of the Royal Society of London*, 186 (Part I):343–424, 1895. 5.4.1

[85] Joelle Pineau, Geoffrey Gordon, and Sebastian Thrun. Anytime point-based approximations for large POMDPs. *Journal of Artificial Intelligence Research*, 27:335–380, 2006. 8.1.2, 8.2.2

[86] Martin L Puterman and Moon Chirl Shin. Modified policy iteration algorithms for discounted markov decision problems. *Management Science*, 24(11):1127–1137, 1978. 9.1

[87] Jad Rahme, Samy Jelassi, Joan Bruna, and S Matthew Weinberg. A permutation-equivariant neural network architecture for auction design. *arXiv preprint arXiv:2003.01497*, 2020. 9.1.2

[88] Jérôme Renault, Eilon Solan, and Nicolas Vieille. Optimal dynamic information provision. *Games and Economic Behavior*, 104:329–349, 2017. 9.1.2

[89] Maytal Saar-Tsechansky and Foster Provost. Handling missing values when applying classification models. *Journal of machine learning research*, 8(Jul):1623–1657, 2007. (b)

[90] Tuomas Sandholm and Anton Likhodedov. Automated design of revenue-maximizing combinatorial auctions. *Operations Research*, 63(5):1000–1025, 2015. 9.1.2

[91] Tuomas Sandholm, Vincent Conitzer, and Craig Boutilier. Automated design of multistage mechanisms. In *IJCAI*, volume 7, pages 1500–1506, 2007. 9.1.2

[92] Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000. 3.4.2, 3.4.2

[93] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 2.2.1, 2.4.3

[94] Weiran Shen, Pingzhong Tang, and Song Zuo. Automated mechanism design via neural networks. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems*, pages 215–223, 2019. 9.1.2

[95] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. Neural Information Processing Systems, 2010. 8.1.2, 8.2.2

[96] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. IJCAI, 2018. 8.1.1

[97] Michael Spence. Job market signaling. *Quarterly Journal of Economics*, 87(3):355–374, 1973. 6.6

[98] Ravi Sundaram, Anil Vullikanti, Haifeng Xu, and Fan Yao. PAC-Learning for Strategic Classification. In *International Conference on Machine Learning (ICML)*, 2021. 1.1

[99] Gregory Valiant and Paul Valiant. An automatic inequality prover and instance optimal identity testing. *SIAM Journal on Computing*, 46(1):429–455, 2017. 6.9

[100] Leslie G Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984. 2.2.1

[101] Fulton Wang and Cynthia Rudin. Falling rule lists. In *Artificial Intelligence and Statistics*, pages 1013–1022, 2015. 4.4

[102] Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. DESPOT: Online POMDP planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017. 8.1.2, 8.2.2

[103] Lan Yu. Mechanism design with partial verification and revelation principle. *Autonomous Agents and Multi-Agent Systems*, 22(1):217–223, 2011. 3.1, 3.1.1, 3.2.1

[104] Hanrui Zhang. Learning Set Functions with Limited Complementarity. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 1.4

[105] Hanrui Zhang. A Generic Truthful Mechanism for Combinatorial Auctions. In *Conference on Web and Internet Economics (WINE)*, 2020. 1.4

[106] Hanrui Zhang. Improved Prophet Inequalities for Combinatorial Welfare Maximization with (Approximately) Subadditive Agents. In *European Symposium on Algorithms (ESA)*, 2020. 1.4

[107] Hanrui Zhang and Vincent Conitzer. A PAC Framework for Aggregating Agents' Judgments. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 1.4

[108] Hanrui Zhang and Vincent Conitzer. Combinatorial Ski Rental and Online Bipartite Matching. In *ACM Conference on Economics and Computation (EC)*, 2020. 1.4

[109] Hanrui Zhang and Vincent Conitzer. Learning the Valuations of a $k$-demand Agent. In *International Conference on Machine Learning (ICML)*, 2020. 1.4

[110] Hanrui Zhang and Vincent Conitzer. Automated Dynamic Mechanism Design. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. (document), 1, 1.3, 1.4

[111] Hanrui Zhang and Vincent Conitzer. Incentive-Aware PAC Learning. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. (document), 1, 1.1, 1.4

[112] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. When Samples Are Strategically Selected. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019. 1.4

[113] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Distinguishing Distributions When Samples Are Strategically Transformed. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019. (document), 1, 1.2, 1.4, 3.1, 7.6

[114] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. When Samples Are Strategically Selected. In *International Conference on Machine Learning (ICML)*, 2019. (document), 1, 1.2, 1.4, 3.1, 4

[115] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Automated Mechanism Design for Classification with Partial Verification. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. (document), 1, 1.1, 1.4, 8.1.2, 8.2.2, 8.4, 9.1.2, 9.4.2, 12

[116] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Classification with Few Tests through Self-Selection. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021. (document), 1, 1.2, 1.4, 3.1

[117] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Efficient Algorithms for Planning with Participation Constraints. In *ACM Conference on Economics and Computation (EC)*, 2022. (document), 1, 1.3, 1.4

[118] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Planning with Participation Constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022. (document), 1, 1.3, 8.2.3, 8.3.1

[119] Hanrui Zhang, Yu Cheng, and Vincent Conitzer. Efficiently solving turn-taking stochastic

games with extensive-form correlation. In *ACM Conference on Economics and Computation (EC)*, 2023. 8.4

[120] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. *ArXiv*, abs/1901.08573, 2019. 1