

POLITECNICO
MILANO 1863

Politecnico di Milano
Dipartimento di Elettronica, Informazione e Bioingegneria
Doctoral Program in Information Technology

Learning Algorithms for Generalized Principal-Agent Problems

Doctoral Dissertation of:
Francesco Bacchiocchi

Supervisor:
Prof. Nicola Gatti

Tutor:
Prof. Francesco Amigoni

The Chair of the Doctoral Program:
Prof. Luigi Piroddi

2025 – Cycle XXXVIII

Abstract

Since the seminal contribution of Myerson [1982], the *generalized principal-agent problem* has become a foundational framework for the analysis of *asymmetric* interactions between *strategic* parties, in which a principal designs incentive schemes to induce the behavior of rational, self-interested agents toward desired outcomes. This framework plays a crucial role in the study of economic interactions in modern digital economies, where platforms and institutions interact on a daily basis with strategic entities with potentially misaligned objectives. Some remarkable examples include *crowdsourcing platforms*, where a central platform must design payment schemes to incentivize workers to complete tasks despite not being able to directly observe the level of effort exerted. Similarly, in *digital advertising* markets, platforms have access to information about users and must decide how to disclose such information to self-interested advertisers whose actions directly affect the revenue generated. Despite its broad applicability, the classical formulation of the generalized principal-agent problem relies on strong assumptions that limit its relevance in practice. Specifically, it requires that the principal fully knows the agents' characteristics as well as the stochastic structure of the environment in order to compute optimal strategies. In practice, such knowledge is not available beforehand. In this thesis, we address these limitations by incorporating *online learning and reinforcement learning* techniques into the classical formulation of the generalized principal-agent problem. We study sequential decision-making settings that allow the principal to operate without prior knowledge of the agents' features, by designing theoretically grounded algorithms that adapt to the environment over time. This approach provides a framework for modelling and analyzing data-driven strategies that better reflect the evolving nature of contemporary digital economic ecosystems, where central agencies and platforms must learn how to influence strategic agents over time.

Contents

1 Introduction **9**
- 1.1 Structure of the work . . . 10
- 1.2 Original contribution . . . 11

2 The generalized principal-agent problem **15**
- 2.1 Stackelberg games . . . 16
- 2.2 Bayesian persuasion . . . 18
- 2.3 Contract design . . . 20

I Learning in Stackelberg Games **23**

3 The sample complexity of Stackelberg games **25**
- 3.1 State of the art of learning in Stackelberg games . . . 26
  - 3.1.1 Existing algorithmic approach . . . 26
  - 3.1.2 Assumptions and limitations . . . 27
  - 3.1.3 Representing leader's strategies with finite precision . . . 28
- 3.2 Learning an optimal strategy to commit to . . . 29
  - 3.2.1 `Learn-Optimal-Commitment` . . . 30
  - 3.2.2 `Find-Hyperplane` . . . 36
  - 3.2.3 `Binary-Search` . . . 43
  - 3.2.4 `Sample-Int` . . . 45
- 3.3 Related works . . . 50

II Learning in Bayesian Persuasion **51**

4 Learning to persuade a single receiver **53**
- 4.1 Online learning in Bayesian persuasion . . . 54
- 4.2 Warm-up: a single signal is all you need . . . 55
- 4.3 Learning to persuade without a clue . . . 57

4.3.1 Phase 1: Build-Search-Space . . . 60
4.3.2 Phase 2: Find-Slices . . . 63
4.3.3 Phase 3: Compute-Signaling . . . 66
4.4 Lower bounds for online Bayesian persuasion . . . 71
4.5 The sample complexity of Bayesian persuasion . . . 75
4.6 Sample complexity with known prior . . . 79
4.7 Related works . . . 82

5 Learning to persuade multiple receivers **83**
5.1 Markov persuasion processes . . . 84
5.2 The learning problem . . . 86
5.2.1 Occupancy measures in MPPs . . . 86
5.2.2 Learning objectives . . . 87
5.2.3 Estimators and confidence bounds . . . 88
5.3 The full feedback case . . . 89
5.3.1 The OPPS algorithm with full feedback . . . 89
5.3.2 Algorithm analysis with full feedback . . . 90
5.3.3 Optimistic optimization problem . . . 98
5.3.4 Lower bound . . . 99
5.4 The partial feedback case . . . 101
5.4.1 The OPPS algorithm with partial feedback . . . 101
5.4.2 Algorithm analysis with partial feedback . . . 102
5.4.3 Lower bound . . . 110
5.5 Related works . . . 112

III Learning in Contract Design **113**

6 Learning bounded contracts **115**
6.1 Learning optimal bounded contracts . . . 116
6.2 The Discover-and-Cover algorithm . . . 118
6.2.1 The Action-Oracle procedure . . . 120
6.2.2 The Try-Cover procedure . . . 129
6.2.3 The Find-HS procedure . . . 136
6.2.4 The Find-Contract procedure . . . 141
6.2.5 Putting it all together . . . 143
6.3 A no-regret algorithm . . . 145
6.4 Related works . . . 146

7 Learning linear contracts **149**
7.1 Bandits with monotone jumps . . . 150
7.1.1 Model . . . 150
7.1.2 Learner's policies and regret . . . 151
7.1.3 No-regret learning in linear contracts . . . 152
7.2 Bandits with monotone jumps applied to auctions . . . 154
7.2.1 No-Regret in posted-price auctions . . . 154
7.2.2 No-regret in first-price auctions . . . 155

7.3 A no-regret algorithm for the BwMJ framework . . . 156
7.3.1 Recursive jump identification with optimistic shrinking . . . 156
7.3.2 The `Find-Jumps` procedure . . . 158
7.3.3 The `Optimistic-Shrink` procedure . . . 161
7.3.4 Putting all together . . . 165
7.4 On the optimality of the `RJI-OS` algorithm . . . 171
7.5 Related works . . . 176

8 Learning robust contracts **177**
8.1 Contract design and approximate best responses . . . 178
8.2 The price of robustness . . . 178
8.3 Online learning of robust contracts . . . 187
8.3.1 Learning interaction . . . 187
8.3.2 A no-regret algorithm . . . 188
8.4 Computational aspects of robust contracts . . . 194
8.5 Related works . . . 198

9 Conclusions and future research **199**

A Additional technical lemmas from Chapter 3 **1**

B Additional technical lemmas from Chapter 4 **5**
B.0.1 The `Find-HS` procedure . . . 6
B.0.2 The `Binary-Search` procedure . . . 9
B.0.3 The `Sample-Int` procedure . . . 9

C Additional technical lemmas from Chapter 5 **15**
C.0.1 Transition probabilities . . . 15
C.0.2 Prior distributions . . . 16
C.0.3 Sender's and receivers' rewards . . . 16

Bibliography **17**

CHAPTER 1

Introduction

In his seminal work, Myerson [1982] introduced the *generalized principal-agent problem* to formalize settings in which one party, *the principal*, aims to incentivize one or more other *agents* to perform desirable actions. Over the decades, this framework has become a crucial tool for describing a wide range of real-world interactions (see, *e.g.*, Gan et al. [2022a]), capturing many *economic scenarios* at the core of today's digital economies, in which a principal must design a mechanism to align self-interested, rational agents toward desired outcomes. In *crowdsourcing platforms*, for instance, a central agency must design payment schemes that incentivize workers to produce high-quality content or perform specific tasks without direct access to the level of effort exerted by the workers. Similarly, in *digital advertising*, ad platforms act as principal and have knowledge of key features of the end users who view the ads, and must decide how to optimally disclose such information to self-interested advertisers. Furthermore, in public policy and regulation, authorities design rules to elicit socially desirable behavior from agents whose private information and motivations remain hidden.

While the classical formulation of the generalized principal-agent problem captures key interactions that arise in today's digital economies, it still relies on a strong assumption, namely that the principal has complete knowledge of the features that define the agent and of the stochasticity of the environment. In many real-world settings, however, the principal lacks such information and therefore cannot design optimal payment or information disclosure mechanisms.

To address these limitations, in this thesis we introduce *online and reinforcement learning* techniques to design sequential decision-making strategies that do not require the principal to have prior knowledge of the agent's features. Integrating

these techniques into the design of such mechanisms makes it possible to build adaptive systems that effectively balance exploration, aimed at gathering information about the environment and the agent, and exploitation, which uses this knowledge to refine and improve the principal's strategy over time. This perspective aims to develop tools that capture the dynamics of modern digital economies, where platforms, regulators, and firms must learn how to influence self-interested users through data-driven strategies.

1.1 Structure of the work

In this section, we describe the structure of the thesis. In Chapter 2, we introduce and formalize the *generalized principal-agent problem*, showing how this framework captures well-known problems in the game-theoretical literature, such as *Stackelberg games*, *Bayesian persuasion*, and *contract design*. For each of these frameworks, we formally present the notation and highlight their distinctive features. The subsequent chapters are organized into three parts, each corresponding to one of these settings, and develop algorithms that relax the assumption that the principal knows all the agent's features in advance. Specifically, Part I revolves around *Stackelberg games* and comprises a single chapter:

- Chapter 3 revises the sample complexity of learning in Stackelberg games, introducing an algorithm that avoids restrictive assumptions of prior work and properly addresses the trade-off between strategy representation precision and the number of samples required to guarantee termination with high probability. The results in this chapter appeared in [Bacchiocchi et al., 2025a] and were published at the *28th International Conference on Artificial Intelligence and Statistics (AISTATS), 2025*.

Part II focuses on *Bayesian persuasion* and is organized into two chapters:

- Chapter 4 studies an online Bayesian persuasion setting where the sender (*i.e.*, the principal) interacts with the receiver (*i.e.*, the agent) over multiple rounds without any knowledge of the prior distribution over the states of nature or the receiver's utilities. In this context, we develop an algorithm that achieves sublinear regret with respect to an optimal signaling scheme, and we complement this result with lower bounds showing the tightness of our approach. These results appeared in [Bacchiocchi et al., 2024a] and were published at the *38th Conference on Neural Information Processing Systems (NeurIPS), 2024*.
- Chapter 5 studies *Markov Persuasion Processes*, in which the sender interacts with a stream of myopic receivers in an unknown Markovian environment. We develop an algorithm that attains sublinear regret with respect to an optimal information-disclosure policy while ensuring that the cumulative violation of persuasiveness is also sublinear, and we complement these guarantees with matching lower bounds. This chapter is based on [Bacchiocchi et al., 2025e], published at the *39th Conference on Neural Information Processing Systems (NeurIPS), 2025*.

Finally, Part III focuses on *contract design* and comprises three chapters:

- Chapter 6 studies the problem of learning approximately optimal *bounded* contracts when the agent's action space is constant. Our main result is an algorithm guaranteed to return an approximately optimal bounded contract in a polynomial number of rounds. Furthermore, we show that this algorithm can be adapted to obtain no-regret guarantees. The results of this chapter are presented in [Bacchiocchi et al., 2024b, 2025b] and were published at the *12th International Conference on Learning Representations (ICLR), 2024*, and in the *Artificial Intelligence Journal (AIJ)*.

- Chapter 7 introduces the *Bandits with Monotone Jumps* framework to study the design of no-regret algorithms for *linear contracts*. This framework captures online learning problems with piecewise-linear reward structures and noisy bandit feedback, including posted-price and first-price auctions. Our main contribution is an algorithm that achieves no-regret guarantees together with a matching lower bound. The results of this chapter are presented in [Bacchiocchi et al., 2025c] and were published at the *26th ACM Conference on Economics and Computation (EC), 2025*.

- Chapter 8 studies the design of contracts under the assumption that the agent plays *approximate best responses*, encompassing scenarios in which the agent has bounded rationality or noisy behavior. First, we provide a no-regret learning algorithm for settings where the principal has no prior knowledge of the key features of the agent. Second, we show that the computation of an optimal principal's contract problem under approximate best responses is solvable in polynomial time. The results of this chapter are presented in [Bacchiocchi et al., 2025d] and were published at the *42nd International Conference on Machine Learning (ICML), 2025*.

1.2 Original contribution

The contributions of this thesis revolve around the design of online and reinforcement learning algorithms in three settings: Stackelberg games, Bayesian persuasion, and contract design. In the following, we briefly outline the contributions for each of these settings.

- **Stackelberg games**. The problem of learning an optimal leader strategy in Stackelberg games without prior knowledge of the follower's utilities was first studied by [Letchford et al., 2009]. Later, [Peng et al., 2019] built on this work to improve its guarantees. However, their approaches assume control over the follower's tie-breaking and a non-degenerate payoffs assumption. Moreover, their guarantees rely on sampling leader strategies uniformly at random, disregarding the precision of their representation.

  In this thesis, we overcome these limitations, providing new algorithmic and analytical tools. Our contribution is threefold. First, we design an algorithm that is guaranteed to work also in degenerate instances and avoids the unrealistic assumptions required in prior approaches. Second, we develop new techniques that explicitly manage the trade-off between sample complexity

and representation precision, ensuring termination with high probability without incurring an exponential dependence on the bit-complexity of the instance. Finally, we show that, with high probability, our algorithm learns an optimal commitment strategy using a number of samples polynomial in the size and bit-complexity of the instance, as long as the action space of either the leader or the follower is fixed, matching the lower bound of [Peng et al., 2019].

- **Bayesian persuasion**. Classical Bayesian persuasion models (see, as example, [Dughmi and Xu, 2016, 2017, Xu, 2020]) rely on strong assumptions, requiring the sender to know both the prior over states of nature and the receiver's utilities. While recent works have used online learning techniques to relax these assumptions, they still presuppose partial knowledge of either the prior [Castiglioni et al., 2020b, 2021b, Bernasconi et al., 2023a] or the receiver's utilities [Zu et al., 2021, Bernasconi et al., 2022, Wu et al., 2022, Lin and Li, 2025]. In this thesis, we consider—to the best of our knowledge, for the first time—settings in which the sender lacks knowledge of both the receivers' payoffs and the prior distribution over states of nature.

  Initially, we study the online learning problem faced by a sender who repeatedly interacts with a receiver, observing the receiver's best-response action after each round as feedback. We provide an explore-then-commit algorithm, based on a non-standard representation of signaling schemes, that achieves regret of order $\widetilde{\mathcal{O}}\left(\binom{n+d}{d}\sqrt{T}\right)$, where T is the learning horizon, d is the number of states of nature, and n is the number of the receiver's actions. We complement our analysis showing that such bound is tight. Our analysis also yields tight bounds on the sample complexity of a PAC-learning version of the problem, where the goal is to identify, with high probability, an approximately optimal signaling scheme in as few rounds as possible.

  We then consider a generalization of the above problem, named *Markov Persuasion Processes* (MPPs), where the sender interacts with a stream of myopic receivers in an unknown Markovian environment. Learning in MPPs without knowledge of receivers' rewards poses additional challenges compared to the setting of [Wu et al., 2022]. Indeed, while a sublinear-regret algorithm that is persuasive at every episode with high probability can be designed in that setting, such a guarantee is not achievable in our case. As a consequence, in this work, we design algorithms that attain sublinear regret while ensuring that the cumulative violation of persuasiveness grows sublinearly with the learning horizon T.

  As a warm-up, we study a full-feedback setting in which, at each round, the sender observes the rewards of all actions in the visited states. We propose an algorithm, *Optimistic Persuasive Policy Search* (`OPPS`), achieving a tight $\mathcal{O}(\sqrt{T})$ regret and violation. We then move to the partial-feedback setting, where the sender only observes the rewards of the actions actually taken in each state. We show that in this setting `OPPS` achieves $\mathcal{O}(T^{\alpha})$ regret and $\mathcal{O}(T^{\max\{\alpha,1-\alpha/2\}})$ violation, where $\alpha \in [1/2, 1]$ is a parameter of the algorithm controlling the amount of exploration.

- **Contract design**. Contract design has received growing attention in the algorithmic game theory community [Dütting et al., 2019, Guruganesh et al., 2021, Dutting et al., 2021, Castiglioni et al., 2022]. However, most works assume that the principal knows the agent's features beforehand and can therefore compute an optimal contract by solving an optimization problem. Only few works relax this assumption by allowing the principal to *learn* these features from experience (see, *e.g.*, [Ho et al., 2014, Zhu et al., 2023, Chen et al., 2024, Cohen et al., 2022, Han et al., 2024a, Duetting et al., 2023a]). In particular, Zhu et al. [2023] propose tight no-regret algorithms for learning optimal bounded and linear contracts in settings with potentially many agent actions. In this thesis, we show that when the agent's action space is small, their guarantees can be significantly improved.

  We start by showing a negative result on the learnability of contracts when the learning space is unbounded. Thus, we restrict our attention to *bounded contracts*, *i.e.*, contracts where payments are bounded by a fixed constant in each outcome and we provide an algorithm that, with high probability, returns an approximately optimal contract in a number of rounds polynomial in the instance size (including the number of outcomes m), when the number of agent's actions n is constant. Furthermore, our algorithm can be slightly modified to achieve cumulative regret—with respect to the optimal bounded contract—bounded by $\widetilde{\mathcal{O}}(m^n\, T^{4/5})$, where $T > 0$ denotes the learning horizon, thereby resolving an open problem posed by Zhu et al. [2023].

  In addition, we study learning algorithms for *linear contracts*, a notable subclass of contracts in which payments depend linearly on the principal's realized reward. To do so, we introduce a general learning framework, named *Bandits with Monotone Jumps* (BwMJ), which provides a unifying perspective for a broad class of online learning problems where the learner's expected utility has a *piecewise-linear* structure satisfying some monotonicity properties. Beyond linear contracts, this framework also covers key problems in microeconomic theory, such as learning in posted-price auctions with finitely many buyer's valuations and first-price auctions under similar assumptions. Our main result is an algorithm with regret $\widetilde{O}(\sqrt{nT})$, where n is the number of "jumps" in the piecewise-linear reward function and T the horizon. This directly yields a $\widetilde{O}(\sqrt{nT})$ no-regret algorithm for linear contracts, where n is the number of agent's actions, improving upon the $\mathcal{O}(T^{2/3})$ bound of Zhu et al. [2023] when $n < T^{1/3}$. Furthermore, in posted-price auction settings, this result improves the state-of-the-art upper bound of Cesa-Bianchi et al. [2019]. Finally, we prove a matching lower bound (up to logarithmic factors), showing the rate is essentially tight.

  Finally, we focus on *robust contracts*, in which the agent may choose actions that are up to $\delta \in (0, 1)$ suboptimal with respect to a best-response action. Contracts designed under this assumption are said to be *robust*, as they are resistant to arbitrary suboptimal behavior of the agent. Within this framework, we first provide upper and lower bounds on the value of an optimal robust contract, *i.e.*, the maximum expected utility the principal can secure

when the agent plays approximate best responses under an adversarial perspective. We then study the learnability of robust contracts in an online learning framework in which the principal and the agent interact over multiple rounds. Our main result is the design of an algorithm that achieves sublinear regret with respect to always playing an optimal robust contract. Our approach presents some advantages compared to the state-of-the-art proposed by Zhu et al. [2023] for the non-robust version of the problem, as it relies on a simpler discretization argument and does *not* require the principal to know its rewards beforehand. Finally, on the computational side, we present a polynomial-time algorithm for computing an optimal robust contract, contrasting with the hardness of analogous commitment problems in Stackelberg games [Gan et al., 2023a].

CHAPTER 2

The generalized principal-agent problem

The *generalized principal–agent problem*, first formulated by Myerson [1982] and later extended by Gan et al. [2022a], models how a *principal* can influence one or more self-interested *agents* toward desired behaviors. As we will further discuss in this chapter, this model captures a wide range of problems in economic and algorithmic game theory literature, such as Stackelberg games, Bayesian persuasion, and contract design.

We consider instances of the generalized principal-agent problem characterized by a finite set of actions A available to the *agent* and by a closed, convex set $X \subseteq \mathbb{R}^d$, which corresponds to the set of strategies available to the *principal*. The principal's and agent's utilities are encoded by two affine functions $u, v : X \times A \to \mathbb{R}$. More formally, for each $a \in A$, both $u(\cdot, a) : X \to \mathbb{R}$ and $v(\cdot, a) : X \to \mathbb{R}$ are affine.

In the one-shot version of the problem, the principal selects a finite set $\Xi \subseteq X$ and a probability distribution $\pi \in \Delta_\Xi$, called the *coordination policy*.[1] In the original formulation of the generalized principal-agent problem by Myerson [1982], the policy π is unconstrained in the sense that it can be any distribution over Ξ without additional restrictions. In contrast, in the broader formulation of the generalized model proposed by [Gan et al., 2022a], the coordination policy may or may not

[1] The model by Gan et al. [2022a] allows the agent to have multiple types and also considers a concave function to encode the principal's utility. Here we restrict the attention to affine functions, which already cover the applications of this thesis. Furthermore, according to Gan et al. [2022a], one should have $\pi \in \Delta_{X \times \mathcal{S}}$, where $\mathcal{S}$ denotes the set of signals. However, since each $x \in \Xi$ can be associated with a unique $s \in \mathcal{S}$ and vice versa, it is without loss of generality to assume $\pi \in \Delta_\Xi$, *i.e.*, that the principal does not randomize over multiple $x \in \Xi$ for a given signal (see, e.g., Lin and Chen [2025]). Furthermore, it is without loss of generality to restrict attention to models where Ξ has finite cardinality (see, e.g., Theorem 1 in Gan et al. [2022a]).

satisfy a set of constraints formally defined as

$$\sum_{x \in \Xi} \pi(x) x \in C$$

for some closed convex set $C \subseteq \mathbb{R}^d$.[2]

The one-shot principal-agent interaction unfolds as follows: (i) the principal commits to a coordination policy $\pi \in \Delta_\Xi$ for some $\Xi \subseteq X$; (ii) a strategy $x \sim \pi$ is sampled according to π and it is communicated to the agent; (iii) the agent selects a *best-response* action $a(x)$ such that $a(x) \in \arg\max_{a \in A} v(x, a)$. As it is customary in the literature, when the agent is indifferent among multiple actions, we assume they select the one that maximizes principal's utility; (iv) the principal achieves utility $u(x, a(x))$ and the agent $v(x, a(x))$. The goal of the principal is to design a coordination policy $\pi \in \Delta_\Xi$ so as to maximize the expected value of their utility, $\sum_{x \in \Xi} \pi(x) u(x, a(x))$.

2.1 Stackelberg games

The Stackelberg paradigm was introduced by von Stackelberg [1934] to describe economic settings where firms compete sequentially. In this framework, one firm, called the leader, moves first by committing to a production strategy, while another firm, called the follower, moves second by reacting to the leader's decision. The paradigm has recently been brought to renewed attention by von Stengel and Zamir [2010], who analyze a variant where the leader commits to a possibly randomized strategy, and the follower best responds after observing it. In recent years, *Stackelberg games* have attracted significant interest, particularly in computer science and economics, as they provide a natural framework to study commitment in strategic environments, finding applications in domains such as security [Paruchuri et al., 2008, Tambe, 2011], toll-setting games [Labbé and Violin, 2016], and network routing [Amaldi et al., 2013].

A Stackelberg game (SG) is defined by a finite set $A_\ell := \{a_i\}_{i=1}^m$ of leader's actions, a finite set $A_f := \{a_j\}_{j=1}^n$ of follower's actions, and utility functions $u_\ell, u_f : A_\ell \times A_f \to [0, 1]$ for the leader and the follower, respectively. The leader commits to a mixed strategy $p \in \Delta_m := \{p \in \mathbb{R}_+^m \mid \sum_{a_i \in A_\ell} p_i = 1\}$, and, after observing it, the follower selects a best response, *i.e.*, an action in

$$A_f(p) := \arg\max_{a_j \in A_f} \sum_{a_i \in A_\ell} p_i u_f(a_i, a_j),$$

breaking ties in favor of the leader [Conitzer and Sandholm, 2006]. That is, the follower plays $a_f^\star(p) \in \arg\max_{a_j \in A_f(p)} \sum_{a_i \in A_\ell} p_i u_\ell(a_i, a_j)$. The leader's goal is to determine a commitment $p \in \Delta_m$ that maximizes their expected utility, namely

$$\max_{p \in \Delta_m} \sum_{a_i \in A_\ell} p_i u_\ell(a_i, a_f^\star(p)).$$

Follower's best responses induce a cover of Δ_m, composed by a family of n best-response regions $\mathcal{P}_j$, defined as follows. Given any pair of follower's actions $a_j, a_k \in$

[2] Given a finite set X, we denote by Δ_X the set of all the probability distributions over X.

A_f such that $a_j \neq a_k$, we let $\mathcal{H}_{jk} \subseteq \mathbb{R}^m$ be the halfspace in which a_j is (weakly) better than a_k in terms of follower's expected utility, where:

$$\mathcal{H}_{jk} \coloneqq \left\{ p \in \mathbb{R}^m \mid \sum_{a_i \in A_\ell} p_i \big(u_f(a_i, a_j) - u_f(a_i, a_k)\big) \geq 0 \right\}.$$

Moreover, we denote by $H_{jk} \coloneqq \partial \mathcal{H}_{jk}$ the hyperplane constituting the boundary of the halfspace $\mathcal{H}_{jk}$, which we call the *separating hyperplane* between a_j and a_k.[3] Then, for every follower's action $a_j \in A_f$, we define its *best-response* region $\mathcal{P}_j \subseteq \Delta_m$ as the subspace of leader's strategies in which action a_j is a best response. The set $\mathcal{P}_j$ is defined as the intersection of Δ_m with *all* the halfspaces $\mathcal{H}_{jk}$ in which a_j is better than another action $a_k \in A_f$. Formally:

$$\mathcal{P}_j \coloneqq \Delta_m \cap \left(\bigcap_{a_k \in A_f : a_k \neq a_j} \mathcal{H}_{jk} \right).$$

The family of all sets $\mathcal{P}_j$ is a cover of Δ_m, since $\Delta_m = \bigcup_{a_j \in A_f} \mathcal{P}_j$. Notice that $\mathcal{P}_j$ is a polytope whose vertices are obtained by intersecting Δ_m with $m-1$ linearly-independent hyperplanes, selected among separating hyperplanes H_{jk} and *boundary hyperplanes* of Δ_m. The latter are defined as $H_i \coloneqq \{p \in \mathbb{R}^m \mid p_i = 0\}$ for every $a_i \in A_\ell$. Since there are at most $n-1$ separating hyperplanes and m boundary ones, the vertices of $\mathcal{P}_j$ are at most $\binom{n+m}{m}$. In the following, we let $V(\mathcal{P}_j) \subseteq \Delta_m$ be the set of all the vertices of $\mathcal{P}_j$, while we denote by $\mathrm{vol}(\mathcal{P}_j)$ its volume relative to Δ_m. Moreover, we let $\mathrm{int}(\mathcal{P}_j)$ be the interior of $\mathcal{P}_j$ relative to Δ_m.[4] Once all the best-response regions are available, the optimization problem faced by the leader to compute an optimal mixed strategy can be formulated as $\max_{a_j \in A_f} \max_{p \in \mathcal{P}_j} u_\ell(p, a_j)$, where the inner $\max$ can be solved efficiently by means of linear programming [Conitzer and Sandholm, 2006].

Stackelberg games can be viewed as a special case of the generalized principal-agent problem, with the leader acting as the principal and the follower as the agent. In this setting the agent's set of actions is $A = A_f$, while the principal's strategy space is $X = \Delta_m$. The principal's and the agent's utility functions are defined as $v(x, a) = \sum_{a_i \in A_\ell} x_i \, u_f(a_i, a)$, and $u(x, a) = \sum_{a_i \in A_\ell} x_i \, u_\ell(a_i, a)$. Notice that both functions are linear for each $a \in A$, as required by the generalized principal-agent problem. In this way, a coordination policy π is a probability distribution over a set of leader mixed strategies $\Xi \subseteq \Delta_\Theta$. However, since an optimal leader mixed strategy always exists, it is without loss of generality to restrict attention to coordination policies π supported on a single element, as is indeed the case in Stackelberg games in which the leader commits to a single mixed strategy.

[3] We let $\partial \mathcal{H}$ be the boundary hyperplane of some halfspace $\mathcal{H} \subseteq \mathbb{R}^m$. Notice that H_{jk} and H_{kj} actually refer to the same hyperplane. In this thesis, we use both names for ease of presentation.

[4] We denote by $\mathrm{vol}_d(\mathcal{P})$ the d-dimensional Hausdorff measure of a polytope $\mathcal{P} \subseteq \mathbb{R}^D$. For ease of notation, when $\mathcal{P} \subseteq \mathbb{R}^m$ and $d = m-1$, we simply write $\mathrm{vol}(\mathcal{P})$. Moreover, we define $\mathrm{int}(\mathcal{P})$ as the interior of $\mathcal{P}$ relative to an affine space of minimal dimension that contains $\mathcal{P}$. In the case of a best-response region $\mathcal{P}_j$, this space is $(m-1)-$dimensional, namely the simplex itself.

2.2 Bayesian persuasion

Bayesian persuasion has been introduced by Kamenica and Gentzkow [2011] to model how strategically disclosing information to decision makers influences their behavior. In recent years, it has attracted considerable attention from the computer science community, since it represents a useful tool for designing algorithms that address strategic interactions among agents with asymmetric information, which are ubiquitous in the real world. As a consequence, Bayesian persuasion has been applied in several settings, such as, *e.g.*, online advertising [Emek et al., 2014, Badanidiyuru et al., 2018, Bacchiocchi et al., 2022, Agrawal et al., 2023], voting [Alonso and Câmara, 2016, Castiglioni et al., 2020a], traffic routing [Vasserman et al., 2015, Bhaskar et al., 2016, Castiglioni et al., 2021a], recommendation systems [Cohen and Mansour, 2019, Mansour et al., 2022], security [Rabinovich et al., 2015, Xu et al., 2016], e-commerce [Miltersen and Sheffet, 2012], and financial regulation [Goldstein and Leitner, 2018].

In its simplest form, Bayesian persuasion involves a *sender* observing some information about the world, called *state of nature*, and a *receiver* who has to take an action. Agents' utilities are misaligned, but they both depend on the state of nature and receiver's action. Thus, sender's goal is to devise a mechanism to (partially) disclose information to the receiver, so as to induce them to take a favorable action. This is accomplished by committing upfront to a *signaling scheme*, encoding a randomized policy that defines how to send informative signals to the receiver based on the observed state.

A Bayesian persuasion instance is characterized by a finite set $\Theta \coloneqq \{\theta_i\}_{i=1}^{d}$ of d states of nature and a finite set $\mathcal{A} \coloneqq \{a_i\}_{i=1}^{n}$ of n receiver's actions. Agents' payoffs are encoded by utility functions $u, u^{\mathrm{s}} : \Theta \times \mathcal{A} \to [0,1]$, with $u_\theta(a) \coloneqq u(\theta, a)$, respectively $u^{\mathrm{s}}_\theta(a) \coloneqq u(\theta, a)$, denoting the payoff of the receiver, respectively the sender, when action $a \in \mathcal{A}$ is played in state $\theta \in \Theta$. The sender observes a state of nature drawn from a *prior* probability distribution $\mu \in \mathrm{int}(\Delta_\Theta)$, with $\mu_\theta \in (0,1]$ denoting the probability of $\theta \in \Theta$. To disclose information about the realized state, the sender can publicly commit upfront to a *signaling scheme* $\phi : \Theta \to \Delta_{\mathcal{S}}$, which defines a randomized mapping from states of nature to signals being sent to the receiver, for a finite set $\mathcal{S}$ of signals. For ease of notation, we let $\phi_\theta \coloneqq \phi(\theta)$ be the probability distribution over signals prescribed by ϕ when the state of nature is $\theta \in \Theta$, with $\phi_\theta(s) \in [0,1]$ denoting the probability of sending signal $s \in \mathcal{S}$.

The sender-receiver interaction goes as follows: (1) the sender commits to a signaling scheme ϕ; (2) the sender observes a state of nature $\theta \sim \mu$ and sends a signal $s \sim \phi_\theta$ to the receiver; (3) the receiver updates their belief over states of nature according to *Bayes rule*; and (4) the receiver plays a best-response action $a \in \mathcal{A}$, with sender and receiver getting payoffs $u_\theta(a)$ and $u^{\mathrm{s}}_\theta(a)$, respectively.

Specifically, an action is a *best response* for the receiver if it maximizes their expected utility given the belief computed in step (3) of the interaction. Formally, given a signaling scheme $\phi : \Theta \to \Delta_{\mathcal{S}}$ and a signal $s \in \mathcal{S}$, we let $\mathcal{A}^\phi(s) \subseteq \mathcal{A}$ be the set of receivers' best-response actions, where:

$$\mathcal{A}^\phi(s) \coloneqq \left\{ a_i \in \mathcal{A} \mid \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(s)(u_\theta(a_i) - u_\theta(a_j)) \geq 0 \quad \forall a_j \in \mathcal{A} \right\}. \tag{2.1}$$

As customary in the literature (see, *e.g.*, [Dughmi and Xu, 2016]), we assume that, when the receiver has multiple best responses available, they break ties in favor of the sender.[5] In particular, we let $a^\phi(s)$ be the best response that is actually played by the receiver when observing signal $s \in \mathcal{S}$ under signaling scheme ϕ, with

$$a^\phi(s) \in \arg\max_{a \in \mathcal{A}^\phi(s)} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(s) u^{\mathrm{s}}_\theta(a).$$

The goal of the sender is to commit to an *optimal* signaling scheme $\phi : \Theta \to \Delta_{\mathcal{S}}$ that maximizes their expected utility, defined as $u^{\mathrm{s}}(\phi) \coloneqq \sum_{s \in \mathcal{S}} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(s) u^{\mathrm{s}}_\theta(a^\phi(s))$. In the following, we let $\mathrm{OPT} \coloneqq \max_\phi u^{\mathrm{s}}(\phi)$ be the optimal value of sender's expected utility. Moreover, given an additive error $\gamma \in (0, 1)$, we say that a signaling scheme ϕ is γ-optimal if $u^{\mathrm{s}}(\phi) \geq \mathrm{OPT} - \gamma$.

In step (3) of the sender-receiver interaction presented above, after observing $s \in \mathcal{S}$, the receiver performs a Bayesian update and infers a posterior belief $\xi^s \in \Delta_\Theta$ over the states of nature, according to the following equation:

$$\xi^s_\theta = \frac{\mu_\theta \, \phi_\theta(s)}{\sum_{\theta' \in \Theta} \mu_{\theta'} \, \phi_{\theta'}(s)} \qquad \forall \theta \in \Theta. \tag{2.2}$$

Consequently, given a signaling scheme ϕ, we can equivalently represent it as a distribution over the set of posteriors it induces. Formally, we say that ϕ induces $\gamma : \Delta_\Theta \to [0, 1]$ if, for each posterior distribution $\xi \in \Delta_\Theta$, we have:

$$\gamma(\xi) = \sum_{s \in \mathcal{S} : \xi^s = \xi} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(s) \qquad \text{and} \qquad \sum_{\xi \in \mathrm{supp}(\gamma)} \gamma(\xi) = 1.$$

Furthermore, we say that a distribution over a set of posteriors γ is consistent, *i.e.*, there exists a valid signaling scheme ϕ inducing γ if the following holds:

$$\sum_{\xi \in \mathrm{supp}(\gamma)} \gamma(\xi) \xi = \mu. \tag{2.3}$$

With an abuse of notation, we will sometimes refer to a consistent distribution over a set of posteriors γ as a signaling scheme. This is justified by the fact that there exists a signaling scheme ϕ inducing such distribution, but we are interested only in the distribution over the set of posteriors that ϕ induces.

To formalize Bayesian persuasion as a special case of the generalized principal-agent problem, we adopt the representation of signaling as probability distributions γ over a set $\Xi \subset \Delta_\Theta$, subject to the constraint defined in Equation (2.3). Here, the sender, acting as the principal, commits to a coordination policy $\pi \in \Delta_\Xi$ with $\Xi \subset \Delta_\Theta$. Thus, to equate the models, the principal's strategy space should satisfy $X = \Delta_\Theta$, and the coordination policy $\pi \in \Delta_\Xi$ must be consistent with the prior μ, *i.e.*, it must satisfy $\sum_{x \in \Xi} \pi(x) \, x = \mu$. This corresponds to a constrained generalized principal-agent problem with constraint set $C = \{\mu\}$. Moreover, by letting the receiver act as the agent, and denoting by $\mathcal{A}$ the set of actions, we can write the utilities of the principal and the agent so that they are both linear for each action $a \in \mathcal{A}$, as required by the model of the generalized principal-agent problem. Formally, $u(x, a) = \sum_{\theta \in \Theta} x_\theta u^{\mathrm{s}}_\theta(a)$ and $v(x, a) = \sum_{\theta \in \Theta} x_\theta u_\theta(a)$ for each action $a \in A$ and $x \in X$.

[5] Given a probability distribution π, we denote its support by $\mathrm{supp}(\pi)$.

2.3 Contract design

Contract theory has been extensively studied in the economic literature [Shavell, 1979, Grossman and Hart, 1992, Holmstrom and Milgrom, 1991] (see the books by Mas-Colell et al. [1995], Bolton and Dewatripont [2005], Laffont and Martimort [2009] for a detailed treatment of the subject). However, the interest in the algorithmics aspects of contract theory have emerged only recently [Dütting et al., 2019, Alon et al., 2021, Guruganesh et al., 2021, Dutting et al., 2021, Dütting et al., 2022, Castiglioni et al., 2022, 2023, Dütting et al., 2023, Guruganesh et al., 2023, Dutting et al., 2024]. Nowadays, thanks to the flourishing of digital economies, contract design finds applications in a terrific number of real-world scenarios, such as, *e.g.*, crowdsourcing platforms [Ho et al., 2014], blockchain-based smart contracts [Cong and He, 2019], and healthcare [Bastani et al., 2016].

In this work, we focus on the *hidden-action principal–agent problem*, a central setting within contract theory where the principal cannot directly observe the agent's actions. In its simplest form, this model describes the interaction between a principal and an agent, where the agent takes a costly action that generates externalities for the principal. In *hidden action* models, the principal observes only the outcome, which is stochastically induced by the agent's action and determines the principal's reward. The principal's objective is to incentivize the agent to choose actions that lead to favorable outcomes. This is achieved by committing to a *contract*, namely a payment scheme that specifies a transfer from the principal to the agent for each possible outcome.

An instance of the hidden-action principal–agent problem is defined by a finite action set $\mathcal{A} = \{a_i\}_{i=1}^{n}$, with $n := |\mathcal{A}|$, available to the agent, and a finite set of possible outcome $\Omega = \{\omega_j\}_{j=1}^{m}$, with $m := |\Omega|$. Each agent's action $a \in \mathcal{A}$ determines a probability distribution $F_a \in \Delta_\Omega$ over outcomes, and it results in a cost $c_a \in [0, 1]$ for the agent. We denote by $F_{a,\omega}$ the probability with which action a results in outcome $\omega \in \Omega$, as prescribed by F_a. Thus, it must be the case that $\sum_{\omega\in\Omega} F_{a,\omega} = 1$ for all $a \in \mathcal{A}$. Each outcome $\omega \in \Omega$ is characterized by a reward $r_\omega \in [0, 1]$ for the principal. Thus, when the agent selects action $a \in \mathcal{A}$, the principal's expected reward is $R_a := \sum_{\omega\in\Omega} F_{a,\omega} r_\omega$.

The principal commits to an outcome-dependent payment scheme with the goal of steering the agent towards desirable actions. Such a payment scheme is called *contract* and it is encoded by a vector $p \in \mathbb{R}_+^m$ defining a payment $p_\omega \geq 0$ from the principal to the agent for each possible outcome $\omega \in \Omega$.[6] Given a contract $p \in \mathbb{R}_+^m$, the agent plays a *best-response* action that is: (i) *incentive compatible* (IC), *i.e.*, it maximizes their expected utility; and (ii) *individually rational* (IR), *i.e.*, it has non-negative expected utility. We assume w.l.o.g. that there always exists an opt-out action $a \in \mathcal{A}$ such that $c_a = 0$ and $R_a = 0$. Such an assumption ensures that there is an action providing the agent with positive utility. This guarantees that any IC action is also IR and allows us to focus w.l.o.g. on IC only.

Whenever the principal commits to $p \in \mathbb{R}_+^m$, the agent's expected utility by playing an action $a \in \mathcal{A}$ is equal to $\sum_{\omega\in\Omega} F_{a,\omega} p_\omega - c_a$, where the first term is the expected payment from the principal to the agent when selecting action a. Then,

[6] As it is customary in contract theory [Carroll, 2015], in this work we assume that the agent has *limited liability*, meaning that the payments can only be from the principal to the agent, and *not viceversa*.

the set $A(p) \subseteq \mathcal{A}$ of agent's best-response actions in a given contract $p \in \mathbb{R}_+^m$ is formally defined as follows:

$$A(p) \coloneqq \left\{ a \in \mathcal{A} \mid a \in \arg\max_{a \in \mathcal{A}} \sum_{\omega \in \Omega} F_{a,\omega} p_\omega - c_a \right\}.$$

Given an action $a \in \mathcal{A}$, we denote with $\mathcal{P}_a \subseteq \mathbb{R}_+^m$ the *best-response set* of action a, which is the set of all the contracts that induce action a as agent's best response. Formally, $\mathcal{P}_a \coloneqq \{p \in \mathbb{R}_+^m \mid a \in A(p)\}$.

As it is customary in the literature (see, *e.g.*, [Dütting et al., 2019]), we assume that the agent breaks ties in favor of the principal when having multiple best responses available. We let $a^\star(p) \in A(p)$ be the action played by the agent in a given $p \in \mathbb{R}_+^m$, which is an action $a \in A(p)$ maximizing the principal's expected utility $\sum_{\omega \in \Omega} F_{a,\omega} (r_\omega - p_\omega)$. For a contract $p \in \mathbb{R}_+^m$ and an action $a \in \mathcal{A}$, the principal's expected utility is $u^{\mathsf{P}}(p, a) \coloneqq \sum_{\omega \in \Omega} F_{a,\omega} \big(r_\omega - p_\omega\big)$, while the agent's expected utility is $u^{\mathsf{A}}(p, a) \coloneqq \sum_{\omega \in \Omega} F_{a,\omega} p_\omega - c_a$. In the classical (single-round) hidden-action principal-agent problem, the goal of the principal is to find a contract $p \in \mathbb{R}_+^m$ that maximizes the expected utility $u^{\mathsf{P}}(p)$. Let $\mathrm{OPT} \coloneqq \max_{p \in \mathbb{R}_+^m} u^{\mathsf{P}}(p)$. A contract p is *optimal* if $u^{\mathsf{P}}(p) = \mathrm{OPT}$, and *$\rho$-optimal* for $\rho > 0$ if $u^{\mathsf{P}}(p) \geq \mathrm{OPT} - \rho$. Furthermore, we also define the *social welfare* as $\mathrm{SW} \coloneqq \max_{a \in \mathcal{A}} R_a - c_a$. With a slight abuse of notation, we also write $u^{\mathsf{P}}(p) \coloneqq \sum_{\omega \in \Omega} F_{a^\star(p),\omega} \big(r_\omega - p_\omega\big)$ to denote the principal's expected utility under contract $p \in \mathbb{R}_+^m$ when the agent selects $a^\star(p) \in \mathcal{A}$.

Finally, we show that hidden-action principal-agent problems can be viewed as a special case of the generalized principal-agent problem. In this case, the agent's set of actions is $A = \mathcal{A}$, while the principal's contract space is $X = \mathbb{R}_+^m$. Thus, for each action $a \in A$, the two functions $u(x, a) = u^{\mathsf{P}}(x, a)$ and $v(x, a) = u^{\mathsf{A}}(x, a)$ are affine in $x \in X$. Finally, similar to the case of Stackelberg games, an optimal coordination policy π is not required to randomize over multiple strategies $x \in X$, since there always exists a contract that maximizes the principal's utility.

Part I

Learning in Stackelberg Games

CHAPTER 3

The sample complexity of Stackelberg games

In this chapter, we study the sample complexity of Stackelberg games, *i.e.*, the number of leader-follower interactions required to learn, with high probability, an optimal leader's mixed strategy. Previous approaches to this problem [Letchford et al., 2009, Peng et al., 2019] rely on restrictive assumptions that limit their applicability. Specifically, they assume a minimum volume of the follower's best-response regions or control over the follower's tie-breaking, and they neglect a key trade-off arising from the fact that leader strategies must be represented with finite precision. As we show in this chapter, if this trade-off is not properly managed, it may require exponentially many samples.

Our algorithm avoids the assumptions made in state-of-the-art approaches and addresses the trade-off between sample complexity and termination probability by introducing new techniques and analyses. In particular, it relies on carefully designed sampling procedures that control the bit-complexity of leader strategies and relax the tie-breaking assumption required by previous approaches by prescribing the leader to commit to suitably defined mixed strategies.

Finally, we prove that with high probability, our algorithm outputs an optimal commitment strategy while using only polynomially many samples in the size and bit-complexity of the instance, provided that either the number of leader's or the follower's actions is fixed. This result confirms the tightness of our analysis, as an exponential dependence on either the number of leader's or the follower's actions is unavoidable [Peng et al., 2019].

3.1 State of the art of learning in Stackelberg games

We study Stackelberg games (SGs) in which the leader does *not* know anything about follower's payoffs and they have to *learn* an optimal strategy to commit to. The leader can only interact with the follower by committing to a strategy $p \in \Delta_m$ and observing the best response $a_f^\star(p)$ played by the latter. We assume that the leader interacts with the follower by calling a function $\texttt{Oracle}(p)$, which takes $p \in \Delta_m$ and returns $a_f^\star(p)$. Since we are concerned with the *sample complexity* of learning an optimal strategy to commit to in SGs, our goal is to design algorithms that the leader can employ to learn such a strategy by using the minimum possible number of *samples*, *i.e.*, calls to $\texttt{Oracle}(p)$. Ideally, we would like algorithms that require a number of samples scaling polynomially in the parameters defining the size of SGs, *i.e.*, the number of leader's actions m, that of follower's ones n, and the number of bits encoding players' payoffs. However, a lower bound by Peng et al. [2019] shows that an exponential dependence in either m or n is unavoidable. Thus, we pursue the goal of learning an optimal strategy to commit to by using a number of samples that is polynomial when either m or n is fixed.

In this chapter, we restrict attention to algorithms operating only on rational numbers. We assume that rational numbers are represented as fractions, specified by two integers encoding the numerator and denominator [Schrijver, 1986]. Given a rational number $q \in \mathbb{Q}$ represented as a fraction b/c with $b, c \in \mathbb{Z}$, we denote the number of bits that q occupies in memory, called *bit-complexity*, as $B_q := B_b + B_c$, where B_b (B_c) is the number of bits required to represent the numerator (denominator). For the sake of the presentation, with an abuse of terminology, given a vector in $\mathbb{Q}^D$ of D rational numbers represented as fractions, we let its bit-complexity be the maximum bit-complexity among its entries. In Appendix A, we present some technical results that will be useful later for dealing with quantities represented with finite precision. Furthermore, in this chapter, we assume that the bit-complexity of the follower's payoff is bounded by a fixed constant. Specifically, we assume that the bit-complexity of each entry of the follower's utility $u_f(a_i, a_j)$, for every pair of actions $a_i \in A_\ell$ and $a_j \in A_f$, is bounded by a fixed constant $L \in \mathbb{N}$.

3.1.1 Existing algorithmic approach

Letchford et al. [2009] first addressed the problem of learning an optimal leader's mixed strategy when the leader has *no* prior knowledge of the follower's payoffs. The main limitation of the approach by [Letchford et al., 2009] is that, in the worst case, it may require a number of samples growing exponentially in the number of leader's actions m and in the representation precision L (expressed in terms of number of bits) of players' payoffs. Peng et al. [2019] later built on the results by Letchford et al. [2009] with the goal of removing the exponential dependence on the payoff precision L, designing an algorithm whose complexity is exponential in either the leader's actions m or the follower's actions n.

A limitation of the algorithm by Peng et al. [2019] is that its theoretical guarantees crucially rely on the assumption that the leader's strategies can be selected uniformly at random *without* taking into account their representation precision. As we will discuss in this chapter, this might lead their algorithm to query an expo-

nential number of samples. Properly accounting for the representation precision of the leader's strategies introduces a challenging trade-off between the number of samples used by the algorithm and its probability of termination.

The core idea underpinning the algorithm by Peng et al. [2019] is to iteratively discover separating hyperplanes H_{jk} to identify all the best-response regions $\mathcal{P}_j$. The algorithm keeps track of some overestimates—called *upper bounds*—of the best-response regions, built by using the separating hyperplanes discovered so far. All the upper bounds are initialized to Δ_m, and, whenever a separating hyperplane H_{jk} is discovered, the upper bounds of both $\mathcal{P}_j$ and $\mathcal{P}_k$ are updated accordingly. Moreover, the algorithm also keeps track of some underestimates—called *lower bounds*—of the best-response regions, with the lower bound of $\mathcal{P}_j$ containing all the points that have already been discovered to belong to $\mathcal{P}_j$. After finding a new separating hyperplane, the algorithm checks each vertex $p \in \Delta_m$ of the (updated) upper bounds, adding it to the lower bound of the best-response region $\mathcal{P}_j$ with $a_j = a_f^\star(p)$ (by taking its convex hull with the lower bound).

In order to find a separating hyperplane, the algorithm by Peng et al. [2019] performs a *binary search* over suitably-defined line segments in Δ_m, by using a procedure introduced by Letchford et al. [2009]. As a first step, the procedure does a binary search on a line segment connecting two points $p^1, p^2 \in \Delta_m$ such that $a_f^\star(p^1) \neq a_f^\star(p^2)$. In particular, the procedure identifies two actions $a_j, a_k \in A_f$ whose upper bounds have an intersection with non-zero volume relative to Δ_m, and it randomly selects p^1 from the interior of the intersection. Moreover, p^2 is any point in the lower bound of either a_j and a_k, chosen so that $a_f^\star(p^1) \neq a_f^\star(p^2)$. The binary search recursively halves the line segment connecting p^1 and p^2 until it finds a point $p^\circ \in \Delta_m$ on some separating hyperplane (this could be potentially different from H_{jk}). Then, the procedure draws a "small" $(m-1)$-dimensional simplex centered at p° uniformly at random, it suitably chooses $m-1$ distinct pairs of vertices of such a simplex, and it performs binary search on the line segments defined by such pairs, finding additional $m-1$ points on the separating hyperplane and identify it.

3.1.2 Assumptions and limitations

The algorithm by Peng et al. [2019] crucially relies on the following assumptions:

(a) each best-response region $\mathcal{P}_j$ is either empty or it has volume $\text{vol}(\mathcal{P}_j)$ greater than or equal to 2^{-nL}, where L is the bit-complexity of follower's payoffs;
(b) when indifferent, the follower breaks ties as needed by the algorithm to correctly terminate;
(c) there are no $m+1$ separating/boundary hyperplanes that intersect in one point; and
(d) no separating hyperplanes coincide.

Assumptions (c) and (d) considerably restrict the set of SGs to which the algorithm applies, and assumption (a) further strengthens these restrictions by excluding cases where a follower's action is a best response only for a small subset of leader's strategies. Assumption (b) is also impractical, as it amounts to giving the algorithm control over which best response the follower plays. The algorithm by Peng et al. [2019] requires (a), (c), and (d) to satisfy the conditions of Letchford

et al. [2009], while (b) is needed for vertex enumeration. As we show later, our algorithm introduces novel techniques to remove assumptions (a)–(d).

While these assumptions already limit the applicability of the algorithm by Peng et al. [2019], further limitations may still arise. In particular, Figure 3.1 shows an instance with $|\mathcal{A}_\ell| = |\mathcal{A}_f| = 3$ that illustrates a structural limitation of this approach, even when all its assumptions are satisfied. As a first step, the algorithm by Peng et al. [2019] prescribes to randomly sample a point from the interior of Δ_m, which in this case coincides with the commitment $p \in \mathcal{P}_3$ in Figure 3.1. Then, it initializes the upper bound $\mathcal{U}_3$ with the whole simplex and queries each of its vertices. When the algorithm queries the vertex v^1, it chooses one of the two followers' best responses and eventually adds the commitment v^1 to a suitably defined lower bound. More specifically, the algorithm sets the lower bound $\mathcal{L}_1 = \{v^1\}$ and also updates its corresponding upper bound $\mathcal{U}_1 = \Delta_3$. Equivalently, it updates the lower bound of the follower's action a_3, *i.e.*, $\mathcal{L}_3 = \text{co}(v^2, v^3, p)$.

As a further step, the algorithm attempts to find a separating hyperplane between the upper bounds $\mathcal{U}_1$ and $\mathcal{U}_3$. To do so, it follows the procedure by Letchford et al. [2009] and deterministically computes the point p^1 in Figure 3.1, lying at the intersection of $\mathcal{U}_1$ and $\mathcal{U}_3 = \Delta_3$. Following the steps in Letchford et al. [2009], the procedure then performs a binary search between the commitment p^1 and the vertex v^1, since $v^1 \in \mathcal{L}_1$ and $a_f^\star(p^1) = a_2$. If the binary search terminates correctly, it returns the vertex v^1 as the point lying on the hyperplane H_{12}. Consequently, at least one vertex of the small simplex centered at v^1 falls outside Δ_3 and cannot be queried, leading to an issue in the procedure by Peng et al. [2019].

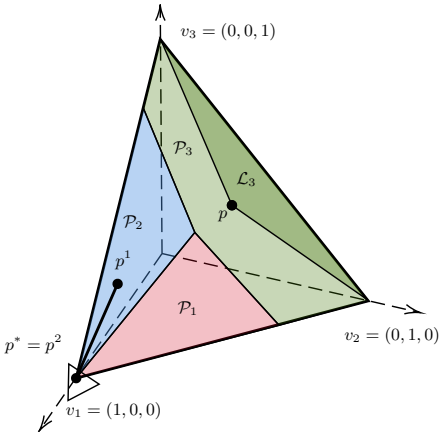

Figure 3.1: *Limitations of the approach by Peng et al. [2019] due to the selection of p^2.*

3.1.3 Representing leader's strategies with finite precision

The analysis by Peng et al. [2019] relies on the assumption that leader's strategies can be selected uniformly at random *without* taking into account their bit-complexity. This allows Peng et al. [2019] to claim that the binary searches performed by their algorithm only require $\mathcal{O}(L)$ samples, where L is the bit-complexity

of follower's payoffs. However, as we show in the rest of this chapter, the number of samples required by those binary searches also depends on the bit-complexity of leader's strategies employed by the algorithm, and, if this is *not* properly controlled, the algorithm by Peng et al. [2019] may require an exponential number of samples. Figure 3.2 shows an example of algorithm execution suffering from the issue described above.

Intuitively, in Figure 3.2, a binary search is performed on the line segment connecting points p^1 and p^2, which are leader's strategies parametrized by some $\epsilon > 0$. The line segment intersects two separating hyperplanes, at points p^3 and p^4, and the distance between p^3 and p^1 can be made arbitrarily small by lowering ϵ. To

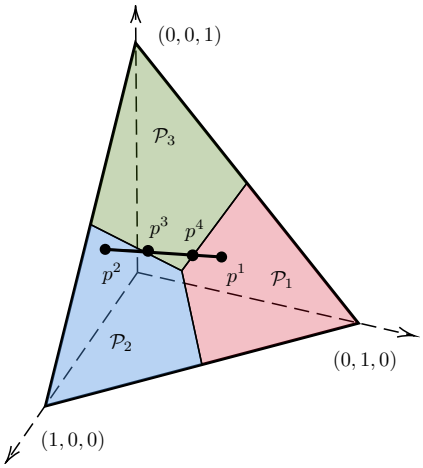

Figure 3.2: *Algorithm by Peng et al. [2019] requires exponentially-many samples.*

identify the separating hyperplane passing through p^4, the binary search procedure must go on until a point on the segment connecting p^3 and p^1 is found. This requires a number of steps logarithmic in the distance between p^1 and p^2 divided by the one between p^1 and p^3. Thus, binary search requires $\mathcal{O}(\log(1/\epsilon))$ samples, and, whenever $\epsilon = \mathcal{O}(\exp(-2^L))$, it requires $\mathcal{O}(2^L)$ samples. Intuitively, the issue arises from the fact that the bit-complexity of the point p^1 is exponential in L. This may happen any time p^1 is randomly selected *without* properly controlling the bit-complexity of leader's strategies. As we show in the rest of this chapter, in order to avoid this issue, it is necessary that leader's strategies have a properly-controlled (finite) bit-complexity. This introduces a trade-off between the number of samples required by the algorithm and its termination probability.

3.2 Learning an optimal strategy to commit to

At a high level, our algorithm works by *closing* follower's actions one after the other, with an action being considered closed when a set of separating hyperplanes identifying its best-response region has been found. The algorithm stops when the union of the identified best-response regions coincides with Δ_m. In this way, it builds all the best-response regions $\mathcal{P}_j$ with $\text{vol}(\mathcal{P}_j) > 0$, which we show to be

sufficient to find an optimal strategy to commit to. To close an action $a_j \in A_f$, our algorithm builds an upper bound $\mathcal{U}_j \subseteq \Delta_m$ of the best-response region $\mathcal{P}_j$, using the separating hyperplanes H_{jk} computed so far. To find a separating hyperplane, the algorithm adopts a binary-search-based procedure similar in nature to the one by Letchford et al. [2009]. However, our procedure has some crucial differences that allow to avoid the issue depicted in Figure 3.2. In particular, it ensures that the point randomly drawn from the intersection of two upper bounds has a properly-controlled bit-complexity. This is accomplished by using a suitable sampling technique, which manages the trade-off between number of samples and termination probability. Moreover, to understand whether action a_j has been closed or *not*, the algorithm checks all the vertices of $\mathcal{U}_j$ to understand if a_j is a best response in each of them, with an idea similar to that in [Peng et al., 2019]. However, to relax assumption (b) made by Peng et al. [2019], our algorithm checks a vertex by querying a nearby point that is in the interior of the upper bound. Such a point is obtained by moving from the vertex towards a known point in the interior of the best-response region. Crucially, provided that the queried point is sufficiently close to the vertex, if a follower's action is a best response in the queried point, then it is also a best response in the vertex, and *viceversa*. Moreover, avoiding querying vertexes also ensures that the failure of the algorithm by Peng et al. [2019] depicted in Figure 3.1 does *not* occur in our algorithm, since it guarantees that a vertex of the currently-considered upper bound is used in a binary search only when it is *not* on the boundary of its associated best-response region.

Next, we describe all the components of our algorithm. Section 3.2.1 introduces `Learn-Optimal-Commitment` which is the main procedure executed by the algorithm. Sections 3.2.2 and 3.2.3 introduce two procedures, `Binary-Search` and `Find-Hyperplane`, respectively, which are designed to work in tandem to find separating hyperplanes. Finally, Section 3.2.4 describes `Sample-Int`, which is called any time the algorithm has to randomly sample a point in the interior of some polytope.

3.2.1 `Learn-Optimal-Commitment`

The pseudocode of `Learn-Optimal-Commitment` is provided in Algorithm 3.1. Notice that the algorithm takes as input a parameter $\zeta \in (0, 1)$, which is used to control the bit-complexity of leader's strategies selected at random, thus managing the trade-off between number of samples and probability of correctly terminating, as shown by our results in the following.[1]

Algorithm 3.1 tracks already-closed follower's actions in a set $\mathcal{C}$. If there are still actions that have to be closed, *i.e.*, $\bigcup_{a_j \in \mathcal{C}} \mathcal{U}_j \neq \Delta_m$, the algorithm identifies one of them by randomly sampling a point from the interior of $\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_k$, containing leader's strategies that have *not* been covered yet by already-found best-response regions (Line 6).With high probability, this sampling step provides a point $p^{\text{int}} \in \Delta_m$ in the interior of some best-response region $\mathcal{P}_j$ with $a_j = \texttt{Oracle}(p^{\text{int}})$ and $a_j \notin \mathcal{C}$. Then, the algorithm focuses on closing action a_j. First, it initializes the up-

[1] For ease of presentation, we assume that there are *no* two actions $a_j, a_k \in A_f$ such that $u_f(a_i, a_j) = u_f(a_i, a_k)$ for all $a_i \in A_\ell$. Indeed, if such two actions exist, their best-response regions $\mathcal{P}_j$ and $\mathcal{P}_k$ coincide, and, thus, in any $p \in \mathcal{P}_j \equiv \mathcal{P}_k$ the best response $a_f^\star(p)$ depends on leader's payoffs (according to tie breaking).

Algorithm 3.1 `Learn-Optimal-Commitment`

Require: Parameter $\zeta \in (0,1)$

1: $B \leftarrow$ Bit-complexity of `Sample-Int` ▷ See Lemma 3.8
2: $\lambda \leftarrow m2^{-m(B+4L)-1}$ ▷ See Lemma 3.1
3: $\delta \leftarrow \zeta / 2(n^2+nm)^2+n^3$
4: $\mathcal{C} \leftarrow \varnothing$ ▷ Set of *closed* follower's actions
5: **while** $\bigcup_{a_j \in \mathcal{C}} \mathcal{U}_j \neq \Delta_m$ **do**
6: $\quad p^{\text{int}} \leftarrow$ Sample a point from $\text{int}\big(\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_k\big)$
7: $\quad a_j \leftarrow \texttt{Oracle}(p^{\text{int}})$
8: $\quad \mathcal{U}_j \leftarrow \Delta_m$ ▷ Initialize upper bound of $\mathcal{P}_j$
9: $\quad \mathcal{V} \leftarrow V(\mathcal{U}_j)$ ▷ Set of unchecked vertices of $\mathcal{U}_j$
10: $\quad$ **while** $\mathcal{V} \neq \varnothing$ **do**
11: $\quad\quad v \leftarrow$ Take any vertex in $\mathcal{V}$
12: $\quad\quad p \leftarrow \lambda p^{\text{int}} + (1-\lambda)v$
13: $\quad\quad a \leftarrow \texttt{Oracle}(p)$
14: $\quad\quad$ **if** $a \neq a_j$ **then**
15: $\quad\quad\quad H_{jk} \leftarrow \texttt{Find-Hyperplane}(a_j, \mathcal{U}_j, p^{\text{int}}, v, \delta)$
16: $\quad\quad\quad \mathcal{U}_j \leftarrow \mathcal{U}_j \cap \mathcal{H}_{jk}$ ▷ Update upper bound
17: $\quad\quad\quad \mathcal{V} \leftarrow V(\mathcal{U}_j)$ ▷ Update unchecked vertices
18: $\quad\quad$ **else**
19: $\quad\quad\quad \mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$ ▷ Vertex v has been checked
20: $\quad\quad$ **end if**
21: $\quad$ **end while**
22: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{a_j\}$ ▷ Action a_j has been closed
23: **end while**
24: $p^\star \leftarrow \arg\max_{p \in \bigcup_{a_j \in \mathcal{C}} V(\mathcal{U}_j)} u_\ell(p)$

per bound $\mathcal{U}_j$ of $\mathcal{P}_j$ to be Δ_m. Then, it iterates over the vertices $V(\mathcal{U}_j)$ of $\mathcal{U}_j$, by employing a set $\mathcal{V}$ containing all the vertices that have *not* been checked yet (Line 9). The algorithm checks a vertex $v \in \mathcal{V}$ by querying a suitable convex combination p of the interior point p^{int} and v (Line 12), to get the best response $a = \texttt{Oracle}(p)$ played by the follower. If the algorithm finds an action $a \neq a_j$, it means that a new separating hyperplane has to be discovered. This is done by calling the `Find-Hyperplane` procedure, which takes as input a_j, the upper bound $\mathcal{U}_j$, the internal point p^{int}, and the vertex v, and it works as described in Section 3.2.2. As shown later, with high probability, the procedure returns a new separating hyperplane H_{jk}, for some action $a_k \in A_f$ possibly different from a. Given H_{jk}, the algorithm updates $\mathcal{U}_j$ by intersecting it with the halfspace $\mathcal{H}_{jk}$ identified by H_{jk}. This may change its vertices, by either adding new ones or removing old ones. Thus, the algorithm re-initializes $\mathcal{V}$ to contain all the vertices $V(\mathcal{U}_j)$ of the new upper bound (Line 17), and it starts checking vertices again.[2] When the algorithm obtains a best response $a = a_j$ in all the vertices in $\mathcal{V}$, it means that all the separating hyperplanes defining the best-response region $\mathcal{P}_j$ have been identified. Thus, it adds a_j to $\mathcal{C}$ and goes on with a follower's action that still has to be closed (if any). A fundamental step of Algorithm 3.1 is to check a vertex $v \in V(\mathcal{U}_j)$ by querying a

[2] The set $\mathcal{V}$ is re-initialized for ease of presentation. It would be possible to update $\mathcal{V}$ without adding back vertices that have already been checked by the algorithm.

nearby point in the interior of $\mathcal{U}_j$. This is obtained by moving towards the interior point p^{int} of $\mathcal{P}_j$. This is crucial to drop assumption (b), needed by the vertex enumeration procedure by Peng et al. [2019]. Indeed, thanks to such an assumption, when the algorithm by Peng et al. [2019] queries a vertex of an upper bound in which there are multiple best responses, it can choose *any* of them as needed in order to complete vertex enumeration. Without assumption (b), `Oracle`(v) for a $v \in V(\mathcal{U}_j)$ may return a follower's action $a_k \neq a_j$ even though action a_j is also a best response in v (due to tie-breaking). Our algorithm circumvents the issue by querying a nearby point in the interior of the upper bound. The following lemma states that such a trick works provided that the point is sufficiently near the vertex. Intuitively, it shows that, if a_j is a best response in a leader's strategy sufficiently close to the vertex towards the interior of $\mathcal{P}_j$, then it is also a best response in the vertex itself, and *viceversa*.

Lemma 3.1. *Given two points $p \in \text{int}(\mathcal{P}_j)$ with $a_j \in A_f$ and $p' \in \Delta_m$, each having bit-complexity bounded by B, let $\widetilde{p} \coloneqq \lambda p + (1-\lambda)p'$ for some $\lambda \in (0, 2^{-m(B+4L)-1})$. Then, $\widetilde{p} \in \mathcal{P}_j \Leftrightarrow p' \in \mathcal{P}_j$.*

Proof. ($\Leftarrow$) If $p' \in \mathcal{P}_j$, then $\widetilde{p}$ is also in $\mathcal{P}_j$ by convexity of $\mathcal{P}_j$.

($\Rightarrow$) In order to prove that $\widetilde{p} \in \mathcal{P}_j$ implies $p' \in \mathcal{P}_j$, we assume by contradiction that $p' \notin \mathcal{P}_j$. This implies that $p' \in \text{int}(\mathcal{H}_{kj} \cap \Delta_m)$ for some follower's action $a_k \in A_f : a_k \neq a_j$, otherwise we would have $p' \in \mathcal{P}_j$ (unless $\mathcal{P}_j = \Delta_m$, but in that case the contradiction is readily obtained). Let $p^\circ \in \Delta_m$ be a leader's strategy defined as $p^\circ \coloneqq \text{co}(p, p') \cap H_{kj}$. Intuitively, p° coincides with the intersection between the separating hyperplane H_{kj} and the line segment connecting the two points p and p'. By definition of p°, there must exist a weight $\lambda^\circ \in (0,1)$ such that $p^\circ = \lambda^\circ p + (1-\lambda^\circ)p'$, since $p^\circ \in \text{co}(p, p')$. Notice that $\lambda^\circ \neq 0$, otherwise we would have $p' \in H_{kj}$, contradicting the assumption that $p' \notin \mathcal{P}_j$. Furthermore, the value of λ° can be easily computed as follows:

$$\lambda^\circ := \frac{\left| \sum_{a_i \in A_\ell} p'_i \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big) \right|}{\left| \sum_{a_i \in A_\ell} (p_i - p'_i) \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big) \right|}.$$

Notice that the numerator in the definition of λ° is the sum of m terms with bit-complexity bounded by $B+4L$, as a consequence of Lemma A.1 in Appendix A. Thus, according to the lemma, the absolute value of their sum is lower bounded by $2^{-m(B+4L)}$, since $\lambda^\circ \neq 0$. Moreover, the denominator of λ° can be upper bounded as follows:

$$\left| \sum_{i=1}^{m} (p_i - p'_i) \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big) \right| \leq 2$$

Thus, the value of λ° can be lower bounded by $2^{-m(B+4L)-1}$. Therefore, by

observing that $\lambda < \lambda^\circ$, we have $\widetilde{p} \in \text{co}(p^\circ, p')$, $\widetilde{p} \neq p^\circ$, and $\widetilde{p} \neq p'$. Thus, $\widetilde{p} \notin \mathcal{P}_j$, since $\widetilde{p} \in \text{int}(\mathcal{H}_{kj} \cap \Delta_m)$. This is the contradiction which concludes the proof. □

The two fundamental properties that hold when Algorithm 3.1 terminates are: (i) each upper bound $\mathcal{U}_j$ with $a_j \in \mathcal{C}$ coincides with $\mathcal{P}_j$, and (ii) all follower's actions $a_j \in A_f$ whose best-response regions have volume $\text{vol}(\mathcal{P}_j) > 0$ have been closed by the algorithm. As we show in the following, Algorithm 3.1 terminates with high probability with properties (i) and (ii) satisfied. The algorithm does *not* terminate whenever either the sampling step in Line 6 does *not* give a point in the interior of $\mathcal{P}_j$ or `Find-Hyperplane` is *not* able to find a new separating hyperplane. The probability of such events happening can be made arbitrarily low by increasing the bit-complexity of the points produced by `Sample-int`, as discussed in Section 3.2.4.

Next, we show that the properties of Algorithm 3.1 are sufficient to find an optimal commitment, by checking all the vertices of upper bounds and taking the one with the highest leader's utility (Line 24). This follows from a property of SGs, which holds since the follower breaks ties in leader's favor. Intuitively, there is always an optimal commitment that is a vertex of a best-response region $\mathcal{P}_j$ with strictly-positive volume.

Lemma 3.2. *Given an SG, there exists an optimal strategy to commit to $p^\star \in \Delta_m$ and a follower's action $a_j \in A_f$ such that $\text{vol}(\mathcal{P}_j) > 0$ and $p^\star = p$ for some vertex $p \in V(\mathcal{P}_j)$ of the best-response region $\mathcal{P}_j$.*

Proof. Let $p^\star \in \Delta_m$ be an optimal strategy to commit to.

First, let us consider the case in which $a_f^\star(p^\star) = a_j$ for some follower's actions $a_j \in A_f$ with $\text{vol}(\mathcal{P}_j) > 0$. In such a case, the statement trivially holds since leader's expected utility is linear in p over the polytope $\mathcal{P}_j$, and, thus, there must exist an optimal commitment (possibly different from $p^\star$) coinciding with one of the vertices of $\mathcal{P}_j$.

Now, let us consider the case in which $a_f^\star(p^\star) = a_k$ for some follower's action $a_k \in A_f$ with $\text{vol}(\mathcal{P}_k) = 0$. As a first step, we show that $p^\star$ is on the boundary (relative to Δ_m) of some best-response region $\mathcal{P}_j$ with $\text{vol}(\mathcal{P}_j) > 0$. Notice that such an action $a_j \in A_f$ always exists, since $\bigcup_{a_k : \text{vol}(\mathcal{P}_k) > 0} \mathcal{P}_k$ coincides with the whole simplex Δ_m.

Suppose by contradiction that $p^\star \in \text{int}(\mathcal{P}_j)$. Then, a_k is a best-response in $\mathcal{P}_j \cap \mathcal{H}_{kj}$, and it is easy to see that this set has positive volume. Hence, we reach a contradiction with $\text{vol}(\mathcal{P}_k) = 0$.

As a consequence, we immediately get that the point $p^\star$ must belong to the boundary (relative to Δ_m) of some $\mathcal{P}_j$ with $\text{vol}(\mathcal{P}_j) > 0$. Moreover, $p^\star$ must also belong to the separating hyperplane H_{jk}, since $\{a_k, a_j\} \subseteq A(p^\star)$. As a result, $p^\star \in \mathcal{P}_j \cap H_{jk}$. Moreover, we have that $\mathcal{P}_j \cap H_{jk} \subseteq \mathcal{P}_k$. This holds since in any point in the set $\mathcal{P}_j \cap H_{jk}$ it must be the case that a_j is a best response, and the utility of actions a_j is equal to the one of action a_k by definition of H_{jk}.

Finally, it is easy to see that $V(\mathcal{P}_j \cap H_{jk}) \subseteq V(\mathcal{P}_j)$ by the definition of $\mathcal{P}_j$.

By the convexity of $\mathcal{P}_j \cap H_{jk}$ and the fact that leader's expected utility is linear over $\mathcal{P}_j \cap H_{jk}$ when follower's best response is fixed to $a_f^\star(p^\star) = a_k$, we can conclude that there exists at least one vertex p of $\mathcal{P}_j \cap H_{jk}$, such that the utility of the leader is equal to the one in $p^\star$ assuming that follower's best response is $a_f^\star(p^\star) = a_k$. The proof is concluded noticing that $a_k \in A(p)$ belongs to the set of best responses, and that ties are broken in favor of the leader. □

Finally, the results of the procedures `Sample-Int` and `Find-Hyperplane` presented in Sections 3.2.2, together with Lemmas 3.1, 3.2, and 3.8, guarantee that we can bound the number of samples required by Algorithm 3.1.

Theorem 3.1. *Given any $\zeta \in (0,1)$, with probability at least $1-\zeta$, Algorithm 3.1 terminates with an optimal strategy to commit to, using a number of samples of order*

$$\widetilde{\mathcal{O}}\left(n^2\left(m^7 L \log\left(\tfrac{1}{\zeta}\right) + \binom{m+n}{m}\right)\right).$$

Proof. We first show that, with probability at least $1-\delta n^2(5(m+n)^2+n)$, Algorithm 3.1 outputs an optimal commitment in finitely many steps. The argument proceeds by induction over the executions of the while loop at Line 10. Specifically, we assume that before an execution of this loop it holds that $\text{vol}(\mathcal{P}_k) > 0$ and $\mathcal{U}_k = \mathcal{P}_k$ for every $a_k \in \mathcal{C}$. Then the loop terminates after finitely many steps, and with probability at least $1-\delta n(5(m+n)^2+n)$ the set $\mathcal{C}$ gains a new action $a_j \notin \mathcal{C}$ with $\text{vol}(\mathcal{P}_j) > 0$ and $\mathcal{U}_j = \mathcal{P}_j$.

To prove the inductive step, we list some useful properties that can be guaranteed (with sufficiently high probability) during the execution of the while loop at Line 10.

1. We first note that, with probability at least $1-\delta n^2$, the commitment p^{int} (sampled at Line 6) belongs to $\text{int}(\mathcal{P}_j)$ for some $a_j \notin \mathcal{C}$ with $\text{vol}(\mathcal{P}_j) > 0$.

   Indeed, p^{int} is drawn from a polytope outside the union of best-response regions $\mathcal{P}_k$ with $a_k \in \mathcal{C}$. By the inductive hypothesis, $\mathcal{P}_k = \mathcal{U}_k$ for all $a_k \in \mathcal{C}$, so $a_f^\star(p^{\text{int}}) \neq a_k$ for all $a_k \in \mathcal{C}$. Moreover, by Lemma 3.8 and a union bound over at most n^2 separating hyperplanes, p^{int} lies in no separating hyperplane with probability at least $1-\delta n^2$. Consequently, p^{int} lies in $\text{int}(\mathcal{P}_j)$ for some $a_j \in A_f$, showing that $\text{vol}(\mathcal{P}_j) > 0$.

2. As a further step, we prove that if $p^{\text{int}} \in \text{int}(\mathcal{P}_j)$ with $\text{vol}(\mathcal{P}_j) > 0$, *i.e.*, point 1 above holds, then with probability at least $1-5\delta n(m+n)^2$, the while loop at Line 10 terminates with at most n iterations.

   To show this, we first notice that as long as Algorithm 3.2 correctly identifies a separating hyperplane whenever invoked, during the execution of such a loop the algorithm never computes the same separating hyperplane H_{jk} with $a_k \in A_f$ more than once. The reason is that whenever Algorithm 3.2 outputs a hyperplane H_{jk}, the upper bound $\mathcal{U}_j$ is intersected with $\mathcal{H}_{jk}$. Moreover, the binary search performed at subsequent steps to determine a point on a (new) separating hyperplane is always executed

either between two interior points of $\mathcal{U}_j$ or between an interior point and a vertex v with $v \notin \mathcal{P}_k$, as guaranteed by Algorithm 3.2 and the check performed at Line 13. Thus, if Algorithm 3.2 correctly identifies a separating hyperplane whenever invoked, then Algorithm 3.2 cannot identify a point lying on an already discovered H_{jk} again, and hence cannot return such a hyperplane.

We now prove that Algorithm 3.2 correctly returns a new separating hyperplane whenever it is invoked during the execution of the loop at Line 10, with probability at least $1 - 5\delta(m+n)^2$. To do so, we employ an inductive argument over the iterations of the loop at Line 13. Specifically, we assume that $\mathcal{U}_j$ is well defined before an iteration, in the sense that it is defined as the intersection of Δ_m and some true separating hyperplanes $H_{jk'}$ for some $a_{k'} \in A_f$. Then, after one iteration of the loop, Algorithm 3.2 either does not compute any separating hyperplane, or computes a true hyperplane H_{jk}, with probability at least $1 - 5\delta(m+n)^2$. First, notice that Algorithm 3.2 is invoked receiving as input a point $p^{\text{int}} \in \text{int}(\mathcal{P}_j)$ such that $\text{vol}(\mathcal{P}_j) > 0$, as ensured with high probability by point 1 above. Furthermore, $v \notin \mathcal{P}_j$, as guaranteed by Lemma 3.6 and by the check at Line 13, while $\mathcal{P}_j \subseteq \mathcal{U}_j$ by the inductive assumption. Thus, the input parameters satisfy the assumptions of Lemma 3.6, which ensures that a hyperplane H_{jk} is correctly computed with probability at least $1 - 5\delta(m+n)^2$.

Finally, using an inductive argument over the iterations of the loop at Line 10, and noting that $\mathcal{U}_j$ initially coincides with the whole simplex Δ_m, we can conclude that during the execution of the loop the hyperplanes computed are true separating hyperplanes defining $\mathcal{P}_j$. Thus, employing a union bound and observing that the number of possible hyperplanes H_{jk} is at most n, with probability at least $1 - 5\delta n(m+n)^2$, we have that the loop terminates in at most n iterations.

3. Finally, we show that if the two points above hold, then after the execution of the loop at Line 10 we have $\mathcal{U}_j = \mathcal{P}_j$. Indeed, throughout the loop, $\mathcal{P}_j \subseteq \mathcal{U}_j$ since $\mathcal{U}_j$ initially coincides with Δ_m and it is intersected with halfspaces $\mathcal{H}_{jk}$. Moreover, when the loop terminates, $\mathcal{U}_j = \text{co}(V(\mathcal{U}_j))$ with $p \in \mathcal{P}_j$ thanks to Lemma 3.1 for every $p \in V(\mathcal{U}_j)$, implying $\mathcal{U}_j \subseteq \mathcal{P}_j$, and thus $\mathcal{P}_j = \mathcal{U}_j$.

As a result, after the while loop at Line 10, and combining the above observations with a union bound, the set $\mathcal{C}$ is updated by adding an action a_k such that $\text{vol}(\mathcal{P}_k) > 0, \mathcal{U}_k = \mathcal{P}_k$, and $a_k \notin \mathcal{C}$, with probability at least $1 - \delta n(5(m+n)^2 + n)$, thereby proving the inductive step. To conclude, we note that the base case of the induction, namely $\mathcal{C} = \varnothing$, follows by the same argument. Hence, when Algorithm 3.1 terminates, the union of the sets $\mathcal{U}_j$ with $a_j \in \mathcal{C}$ coincides with Δ_m with probability at least $1 - \delta n^2(5(m+n)^2 + n)$. By Lemma 3.2, Algorithm 3.1 returns an optimal strategy to commit to in finitely many steps with probability at least $1 - \delta n^2(5(m+n)^2 + n)$.

Finally, we observe that, with the same probability as above, the number of samples required by Algorithm 3.1 is

$$\mathcal{O}\left(n^2\left(m^7 L \log\left(\tfrac{1}{\delta}\right) + \binom{m+n}{m}\right)\right).$$

Indeed, during the loop at Line 10, Algorithm 3.2 is invoked at most n times, and each invocation requires $\mathcal{O}\left(m^7 L \log\left(\tfrac{1}{\delta}\right)\right)$ samples by Lemma 3.6. Moreover, before computing a hyperplane, in the worst case Algorithm 3.1 queries every vertex of $\mathcal{U}_k$. Since each $\mathcal{U}_k$ has at most $\binom{m+n}{m}$ vertices and this check is performed at most n times, the total number of samples used in the loop is $\mathcal{O}\left(n\left(m^7 L \log\left(\tfrac{1}{\delta}\right) + \binom{m+n}{m}\right)\right)$. As the number of follower actions is n, the loop at Line 10 is executed at most n times. Hence, the overall sample complexity of Algorithm 3.1 is $\mathcal{O}\left(n^2\left(m^7 L \log\left(\tfrac{1}{\delta}\right) + \binom{m+n}{m}\right)\right)$. Setting $\zeta = \delta n^2(5(m+n)^2+n)$, we conclude that with probability at least $1-\zeta$, the sample complexity is

$$\widetilde{\mathcal{O}}\left(n^2\left(m^7 L \log\left(\tfrac{1}{\zeta}\right) + \binom{m+n}{m}\right)\right). \qquad \square$$

We observe that, by Theorem 3.1, the number of samples used by Algorithm 3.1 is polynomial when fixing either the number of leader's actions m or that of follower's actions n.

3.2.2 `Find-Hyperplane`

The pseudocode for the `Find-Hyperplane` procedure is in Algorithm 3.2. Algorithm 3.1 ensures that it gets as input an action $a_j \notin \mathcal{C}$ to be closed, the upper bound $\mathcal{U}_j$ of $\mathcal{P}_j$, an interior point $p^{\text{int}} \in \text{int}(\mathcal{P}_j)$, a vertex $v \in V(\mathcal{U}_j)$ with $v \notin \mathcal{P}_j$, and the parameter $\delta \in (0,1)$. Algorithm 3.2 samples a point p in the interior of $\mathcal{U}_j$, using Algorithm 3.4. If $\texttt{Oracle}(p) = a_j$, then it performs binary search between p and the vertex v. Instead, if $\texttt{Oracle}(p)$ returns an action $\neq a_j$, then it performs binary search between p^{int} and p. Selecting p at random is fundamental to ensure that, with high probability, the segment on which binary search is performed does *not* intersect a point on a facet of $\mathcal{P}_j$ where two or more separating hyperplanes H_{jk} intersect. Moreover, by Lemma 3.1 and $\texttt{Oracle}(v) \neq a_j$, action a_j is *never* a best response in v. This excludes that binary search is done on a segment fully contained in $\mathcal{P}_j$, an issue of Peng et al. [2019] (see Figure 3.1). After the two extremes p^1, p^2 of the segment have been identified, Algorithm 3.2 calls the `Binary-Search` procedure (Line 8). The latter computes a point $p^\circ \in \Delta_m$ on a new separating hyperplane H_{jk} with $a_k \in A_f$. However, to actually compute the hyperplane, the algorithm has to identify $m-1$ linearly-independent points on it.[3] To do so, it uses a method different from the one implemented in existing algorithms. Instead of randomly drawing a "small" $(m-1)$-dimensional simplex centered at p°, our algorithm samples a point from the interior of each facet $H_i \cap \Delta_m$ of leader's strategy space, by calling `Sample-Int` (Line 12).[4] Then, it takes suit-

[3] Notice that $m-1$ linearly-independent points are sufficient since separating hyperplanes pass by the origin.

[4] When $m = 2$, the loop at Line 11 can be skipped, as the algorithm can simply employ p° to compute H_{jk}. Furthermore, notice that a facet $H_i \cap \Delta_m$ of Δ_m can be mapped one-to-one to a $(d-1)$-dimensional simplex Δ_d with $d = m-1$, on which the `Sample-Int` procedure (Algorithm 3.4) can be applied.

able linear combinations between sampled points and p°, defined by a parameter α (see Line 9). Specifically, the algorithm takes the points $p^{+i} := p^\circ + \alpha(p - p^\circ)$, where p is the point sampled from the facet $H_i \cap \Delta_m$. Moreover, the algorithm also takes the points $p^{-i} := p^\circ - \alpha(p - p^\circ)$, which represent "mirrored" versions of the points p^{+i}. Algorithm 3.2 splits points p^{+i} and p^{-i} into two sets $\mathcal{S}_j$ and $\mathcal{S}_k$, depending on which follower's action is a best response in such points, either a_j or a_k. Considering the "mirrored" points p^{-i} is needed to ensure that the algorithm takes at least one point in which follower's best response is a_j and at least one point in which the best response is a_k.[5] Finally, Algorithm 3.2 takes $m - 1$ pairs of linearly-independent points, by taking one point from $\mathcal{S}_j$ and the other from $\mathcal{S}_k$, and it computes the coefficients of the separating hyperplane H_{jk}. Figure 3.3 shows an example of how Algorithm 3.2 works.

Before presenting the theoretical guarantees of Algorithm 3.2, we introduce a set of assumptions on its input values, which hold with high probability thanks to Algorithm 3.1. Formally:

Assumption 3.1. *Algorithm 3.2 receives as input an action $a_j \in A_f$, a vertex $v \notin \mathcal{P}_j$, a point $p^{\text{int}} \in \text{int}(\mathcal{P}_j)$ with bounded bit complexity, and a region $\mathcal{U}_j$ such that $\mathcal{P}_j \subseteq \mathcal{U}_j$ and $\text{vol}(\mathcal{P}_j) > 0$.*

Algorithm 3.2 `Find-Hyperplane`

Require: $a_j, \mathcal{U}_j, p^{\text{int}}, v, \delta$ ▷ As given by Algorithm 3.1
1: $p \leftarrow$ `Sample-Int`$(\mathcal{U}_j, \delta)$
2: $p^1 \leftarrow p$ ▷ p^1 always selected at random
3: **if** `Oracle`$(p) = a_j$ **then**
4: $\quad p^2 \leftarrow v$ ▷ $v \notin \mathcal{P}_j$ by design
5: **else**
6: $\quad p^2 \leftarrow p^{\text{int}}$ ▷ `Oracle`$(p) \neq a_j$
7: **end if**
8: $p^\circ \leftarrow$ `Binary-Search`(a_j, p^1, p^2)
9: $\alpha \leftarrow 2^{-m(B+4L)-1}/m$ ▷ $B =$ bit-complexity of p°
10: $\mathcal{S}_j \leftarrow \varnothing; \mathcal{S}_k \leftarrow \varnothing$
11: **for** $i = 1 \dots m$ **do**
12: $\quad p \leftarrow$ `Sample-Int`$(H_i \cap \Delta_m, \delta)$ ▷ Sample a facet
13: $\quad p^{+i} \leftarrow p^\circ + \alpha(p - p^\circ)$
14: $\quad p^{-i} \leftarrow p^\circ - \alpha(p - p^\circ)$
15: $\quad$ **if** `Oracle`$(p^{+i}) = a_j$ **then**
16: $\quad\quad \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{p^{+i}\} \wedge \mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{p^{-i}\}$
17: $\quad$ **else**
18: $\quad\quad \mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{p^{+i}\} \wedge \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{p^{-i}\}$
19: $\quad$ **end if**
20: **end for**
21: Build H_{jk} by `Binary-Search`(a_j, p^1, p^2) for $m-1$ pairs of linearly-independent points $p^1 \in \mathcal{S}_j, p^2 \in \mathcal{S}_k$

Under Assumption 3.1, we prove that the point p° computed by Algorithm 3.2

[5] Indeed, it would be sufficient to take a single point p^{-i}, and only in the case in which all points p^{+i} induce the same best response. Algorithm 3.2 computes all the p^{-i} for ease of presentation.

belongs, with high probability, to a single hyperplane H_{jk}.

Lemma 3.3. *Under Assumption 3.1, with probability at least* $1 - 3\delta(m+n)^2$*, the point* p° *belongs to* $\text{int}(\Delta_m)$ *and lies on a single hyperplane* H_{jk} *with* $\text{vol}(\mathcal{P}_k) > 0$*.*

Proof. We first show that, with probability $1-n\delta$, the point p^1 selected at random in Algorithm 3.2 does not belong to any hyperplane H_{jk} with $a_k \in A_f$. This follows from Algorithm 3.4, since, by assumption, $\text{vol}(\mathcal{U}_j) \geq \text{vol}(\mathcal{P}_j) > 0$, and by applying a union bound over the follower's actions. Therefore, the point p^1 either belongs to $\text{int}(\mathcal{P}_j)$ or satisfies $p^1 \notin \mathcal{P}_j$. If $p^1 \in \text{int}(\mathcal{P}_j)$, then $a_f^\star(p^1) = a_j$, so $p^2 = v$, and, by assumption, $v \notin \mathcal{P}_j$. On the other hand, if $a_f^\star(p^1)$ is different from a_j, then $p^1 \notin \mathcal{P}_j$, while $p^2 = p^{\text{int}}$ belongs to $\text{int}(\mathcal{P}_j)$ by assumption. Furthermore, since the bit complexities of both p^1 and p^2 are bounded, Algorithm 3.3 returns a point $p^\circ \in \text{int}(\Delta_m)$ on some H_{jk} with $a_k \in A_f$.

We prove now that the point p° lies on a $(m-2)-$dimensional face of $\mathcal{P}_j$ with high probability. Let H and H' be two distinct hyperplanes whose corresponding halfspaces define $\mathcal{P}_j$. We bound the probability that the line segment connecting p^1 to p^2 does not intersect any non empty region $\mathcal{P} := H \cap H'$. Since H and H' are two distinct hyperplanes, the set $\mathcal{P}$ is an affine subspace of dimension at most $m-2$. Let

$$H'' := \text{aff}\{p^2, \mathcal{P}\},$$

be the smallest affine subspace that contains both p^2 and $\mathcal{P}$. Then H'' has dimension at most $m-1$. We notice that $p^2 \in \{v, p^{\text{int}}\}$. At this point, we must bound the probability that $p^1 \notin H''$, for any possible choice of the point p^2 and affine subspace H'' defined above. This probability is at least $1-2\delta(m+n)^2$. This follows from Lemma 3.8 together with a union bound over all possible pairs of hyperplanes H and H' defining the facets of $\mathcal{P}_j$, which are at most $\binom{m+n}{2} \leq (m+n)^2$, and over the two possible choices of p^2.

We remark that Lemma 3.8 applies since $\text{vol}(\mathcal{U}_j) > 0$ by assumption and Δ_m is not contained in H'', since the origin belongs to H'' but not to the simplex. Therefore, with probability at least $1-2\delta(m+n)^2$, the segment connecting p^1 to p^2 intersects $\partial\mathcal{P}_j$ in exactly one point, which lies in the relative interior of a facet of $\mathcal{P}_j$. Thus, p° lies on a unique facet of $\mathcal{P}_j$. Consequently, there exists a unique hyperplane H_{jk} such that $p^\circ \in H_{jk}$. Notice that such a hyperplane is unique, since if two separating hyperplanes H_{jk} and $H_{jk'}$ had an $(m-2)$-dimensional nonempty intersection containing the facet of $\mathcal{P}_j$ that includes p°, and both contained the origin, this would imply that the two hyperplanes coincide and that the follower's actions a_k and $a_{k'}$ are identical, which is not possible under the assumption that the follower's actions are distinct. Furthermore, there always exists an m-dimensional hypersphere centered at p° whose intersection with $\mathcal{H}_{kj}$ has strictly positive volume. Thus, $\text{vol}(\mathcal{P}_k) > 0$.

The proof is concluded by taking a union bound over the event that p^1 does not belong to any separating hyperplane H_{jk}, and that the line segment connecting p^1 to p^2 intersects the boundary of $\mathcal{P}_j$ in a single facet. □

We now discuss two crucial properties of points p^{+i}:

Lemma 3.4. *Under Assumption 3.1, if the bit-complexity of p° is bounded by B, then with probability at least $1-2\delta m-3\delta(m+n)^2$, the points p^{+i} computed by Algorithm 3.2 satisfy the following: (i) they lie in* int(Δ_m)*, (ii) they are linearly independent, and (iii) they are* not *on H_{jk}, where H_{jk} is the hyperplane such that $p^\circ \in H_{jk}$.*

Proof. First, we observe that $p^\circ \in \text{int}(\Delta_m)$ thanks to Lemma 3.3. We prove the following three properties.

1. We show that each p^{+i} with $i \in [m]$ belongs to int(Δ_m). Indeed,

$$\sum_{j=1}^{m} p_j^{+i} = \sum_{j=1}^{m} \left(p_j^\circ + \alpha(p_j - p_j^\circ)\right) = \sum_{j=1}^{m} p_j^\circ + \alpha \sum_{j=1}^{m} (p_j - p_j^\circ) = 1,$$

   so $p^{+i} \in \Delta_m$. Moreover, for every $j \in [m]$,

$$p_j^{+i} = p_j^\circ + \alpha(p_j - p_j^\circ) \geq \frac{1}{2^B} - \frac{1}{m2^{m(B+4L)-1}} > 0.$$

   The inequality holds because $p_j^\circ \geq 2^{-B}$ for all $j \in [m]$, by the bit-complexity bound on p°, and because $\alpha(p_j - p_j^\circ) \geq -\alpha$ for all j. Therefore, every coordinate of p^{+i} is strictly positive. Hence $p^{+i} \in \text{int}(\Delta_m)$ for all $i \in [m]$.

2. We prove that, with probability at least $1-\delta m$, the points p^{+i}, $i \in [m]$, computed by Algorithm 3.2 are linearly independent. For $k \in [m]$, let

$$H'_k := \text{span}\{p^{+1}, \ldots, p^{+k}\}.$$

   It suffices to show that $p^{+(k+1)} \notin H'_k$ for all $k \in [m-1]$. Since

$$p^{+(k+1)} = (1-\alpha)p^\circ + \alpha p$$

   with p sampled from the facet $H_{k+1} \cap \Delta_m$, we have

$$p^{+(k+1)} \in H'_k \iff p \in H''_k := \left\{\lambda_l \in \mathbb{R} \mid \sum_{l=1}^{k} \lambda_l\, p^{+l} + \left(1 - \tfrac{1}{\alpha}\right)p^\circ\right\},$$

   so H''_k is an affine subspace with $\dim(H''_k) \leq k$.

   We prove that, for every $k \in [m-1]$, it holds $H_{k+1} \cap \Delta_m \not\subset H''_k$. We split the proof into three cases.

   Case $k \leq m-3$. We have $\dim(H''_k) \leq k \leq m-3$, while $\dim(H_{k+1} \cap \Delta_m) = m-2$. A set of dimension $m-2$ cannot be contained in one of dimension at most $m-3$. Hence $H_{k+1} \cap \Delta_m \not\subset H''_k$.

   Case $k = m-2$. Suppose, by contradiction, that $F := H_{m-1} \cap \Delta_m \subseteq H''_{m-2}$. Then, we have:

$$F = F \cap \left\{x \in \mathbb{R}^m \mid \sum_{i=1}^{m} x_i = 1\right\} \subseteq H''_{m-2} \cap \left\{x \in \mathbb{R}^m \mid \sum_{i=1}^{m} x_i = 1\right\}.$$

We notice that $H''_{m-2} \not\subseteq \left\{x \in \mathbb{R}^m \mid \sum_{i=1}^m x_i = 1\right\}$. Thus:

$$\dim\left(H''_{m-2} \cap \left\{x \in \mathbb{R}^m \mid \sum_{i=1}^m x_i = 1\right\}\right) \leq m-3.$$

This is a contradiction, since $\dim(F) = m-2$. Therefore, $F \not\subseteq H''_{m-2}$.

Case $k = m-1$. Suppose, by contradiction, that $F := H_m \cap \Delta_m \subseteq H''_{m-1}$. Notice that F lies in the simplex hyperplane defined as follows:

$$\Delta_m^+ := \left\{x \in \mathbb{R}^m \mid \sum_{i=1}^m x_i = 1\right\},$$

so $F \subseteq H''_{m-1} \cap \Delta_m^+$. Now H''_{m-1} is an affine hyperplane of dimension $m-1$. Since $H''_{m-1} \not\subseteq \Delta_m^+$, their intersection has dimension $m-2$.

On the other hand, F is defined inside Δ_m^+ by the additional constraint $x_m = 0$, so it has dimension $m-2$. If F were contained in $H''_{m-1} \cap \Delta_m^+$, we would have equality of two affine subspaces of the same dimension. But $H''_{m-1} \cap \Delta_m^+$ contains points with $x_m > 0$, while every point in F satisfies $x_m = 0$. Hence equality is impossible, thus $F \not\subseteq H''_{m-1}$.

In all cases $H_{k+1} \cap \Delta_m \not\subset H''_k$. Thus, employing Lemma 3.8, we have

$$\mathbb{P}\big(p \in H''_k \cap (H_{k+1} \cap \Delta_m)\big) \leq \delta$$

for each $k \in [m-1]$.

By a union bound over $k = 1, \ldots, m-1$, we conclude that all p^{+i}, with $i \in [m]$, are linearly independent with probability at least $1 - \delta m$.

3. To conclude the proof, we bound the probability that each point p^{+i} does not belong to the separating hyperplane H_{jk}. We observe that p^{+i} lies in $H_{jk} \cap \Delta_m$ only if the point p sampled from $H_i \cap \Delta_m$ also lies in H_{jk}. This is because p^{+i} lies on the segment connecting p to p° and $p^\circ \in H_{jk}$ thanks to Lemma 3.3 with probability at least $1 - 3\delta(m+n)^2$.

   By Lemma 3.8, Algorithm 3.4 guarantees that the probability of sampling $p \in H_{jk} \cap (H_i \cap \Delta_m)$ is at most δ, since $H_i \cap \Delta_m \not\subseteq H_{jk}$ for every $i \in [m]$. Applying a union bound over all m facets, we conclude that $p^{+i} \notin H_{jk}$ for all $i \in [m]$ with probability at least $1 - \delta m$.

Finally, by employing a union bound, we have that the lemma holds with probability at least $1 - 2\delta m - 3\delta(m+n)^2$. □

Lemma 3.4 is proved by means of `Sample-Int`, shown in Lemma 3.8. The following lemma shows that the sets $\mathcal{S}_j$ and $\mathcal{S}_k$ built by Algorithm 3.2 are always well defined. It shows that, in each point p^{+i} or p^{-i}, either a_j or a_k is a best response for the follower, and, additionally, the best response in p^{+i} is always different from that in p^{-i}. Formally:

Lemma 3.5. *Under Assumption 3.1, if the bit-complexity of p° is bounded by B, then with probability at least $1 - 3\delta(m+n)^2 - 2\delta m$, if any p^{+i} computed in Algorithm 3.2 does not belong to H_{jk}, then $p^{+i} \in \text{int}(\mathcal{P}_j)$ or $p^{+i} \in \text{int}(\mathcal{P}_k)$. Moreover, if $p^{+i} \in \text{int}(\mathcal{P}_j)$, then $p^{-i} \in \text{int}(\mathcal{P}_k)$, and vice versa.*

Proof. Since p° belongs to a single hyperplane H_{jk} with $\text{vol}(\mathcal{P}_k) > 0$, thanks to Lemma 3.3 with probability at least $1 - 3\delta(m+n)^2$, we have that there exists a hypersphere centered at p° that is fully contained in $\text{int}(\mathcal{P}_j \cup \mathcal{P}_k)$.

Formally, we define

$$\mathbb{S}_\epsilon^m(p^\circ) \coloneqq \{x \in \mathbb{R}^m \mid \|x - p^\circ\|_2 \leq \epsilon\}$$

as the sphere in $\mathbb{R}^m$ of radius ϵ centered in p° and we show that when $\epsilon < \epsilon' = 2^{-m(B+4L)}/m$ such a sphere has null intersection with the boundaries of both $\mathcal{P}_j$ and $\mathcal{P}_k$. To do that, we prove that the $\|\cdot\|_2-$distance between the commitment p° and any separating hyperplane defining the facets of both $\mathcal{P}_j$ and $\mathcal{P}_k$ can be computed as follows:

$$d_2(p^\circ, H_{lm}) = \left| \frac{\sum_{i=1}^{m} p_i^\circ \Big(u_f(a_i, a_l) - u_f(a_i, a_m)\Big)}{\sqrt{\sum_{i=1}^{m} \Big(u_f(a_i, a_l) - u_f(a_i, a_m)\Big)^2}} \right| \geq \frac{1}{m 2^{m(B+4L)}}$$

for every

$$(l, m) \in \bigcup_{k' \in [n]} (k', j) \cup \bigcup_{j' \in [n]} (k, j')$$

such that H_{lm} does not coincide with H_{jk}. We observe that the inequality follows by means of Lemma A.1, as the numerator of the above fraction is the sum of m terms whose bit complexities are bounded by $B + 4L$. Furthermore, the denominator of such a quantity is at most m. Analogously, we can prove that the same lower bound holds for each boundary plane H_i with $i \in [m]$, defining the facets of either $\mathcal{P}_j$ or $\mathcal{P}_k$.

We also show that all the commitments p^{+i} with $i \in [m]$ computed in Algorithms 3.2 are such that $p^{+i} \in \mathbb{S}_\epsilon^m(p^\circ)$ with

$$\epsilon < \epsilon' = \frac{2^{-m(B+2L)}}{m}.$$

Formally, we have:

$$\|p^{+i} - p^\circ\|_2 = \|p^\circ + \alpha(p - p^\circ) - p^\circ\|_2 = \alpha\|p^\circ - p\|_2 \leq 2^{-m(B+4L)-1/2}/m < \epsilon'.$$

The first equality holds because of the definition of p^{+i}, while the second inequality holds because $\max_{p,p' \in \Delta_m} \|p - p'\|_2 \leq \sqrt{2}$ and thanks to the definition of α in Algorithm 3.2. Thus, observing that each $p^{+i} \in \mathbb{S}_\epsilon^m$ for each $i \in [m]$, and

that p^{+i} is not on H_{jk}, we have that in each p^{+i} either a_j or a_k is the follower's unique best response. Finally, we show that if $a_f^\star(p^{+i}) = a_j$, then $a_f^\star(p^{-i}) = a_k$. More specifically, if $a_f^\star(p^{+i}) = a_j$ and $p^{+i} \notin H_{jk}$, then the following holds:

$$\sum_{i=1}^{m} p_i^{+i}\Big(u_f(a_i,a_j) - u_f(a_i,a_k)\Big) = \alpha \sum_{i=1}^{m} (p_i^\circ - p_i)\Big(u_f(a_i,a_j) - u_f(a_i,a_k)\Big) > 0.$$

This, in turn, implies that:

$$0 > -\alpha \sum_{i=1}^{m} \Big(p_i^\circ - p_i)(u_f(a_i,a_j) - u_f(a_i,a_k)\Big) = \sum_{i=1}^{m} p_i^{-i}\Big(u_f(a_i,a_j) - u_f(a_i,a_k)\Big),$$

since $p^\circ \in H_{jk}$ by Lemma 3.3. Finally, with the same argument, we can show that if $a_f^\star(p^{+i}) = a_k$, then $a_f^\star(p^{-i}) = a_j$, concluding the proof. □

Lemma 3.5 holds since there exists a neighborhood of p° that does *not* intersect any other separating/boundary hyperplane defining $\mathcal{P}_j$ and $\mathcal{P}_k$, as the bit-complexity of all elements involved is bounded. Moreover, with high probability, the segment on which binary search is performed does *not* intersect hyperplanes different from H_{jk}. By Lemma 3.5, with high probability, the sets $\mathcal{S}_j$ and $\mathcal{S}_k$ are well

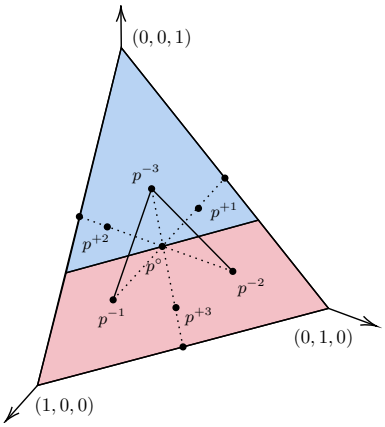

Figure 3.3: *Example of points p^{+i} and p^{-i} computed by Algorithm B.1, together with the $m-1$ line segments on which the algorithm performs binary search to compute the hyperplane H_{jk}.*

defined, namely, it is possible to identify $m-1$ pairs p^1, p^2 of linearly-independent points with $p^1 \in \mathcal{S}_j$ and $p^2 \in \mathcal{S}_k$. Finally, by Lemmas 3.4 and 3.5, we have:

Lemma 3.6. *With probability at least $1 - 3(m+n)^2\delta - 2\delta m$, Algorithm 3.2 returns a separating hyperplane H_{jk} by using at most $\mathcal{O}(m^7 L + m^4 \log(1/\delta))$ samples.*

Proof. To prove the lemma we observe that all the points p^{+i} with $i \in [m]$ are linearly independent and do not belong to the separating hyperplane H_{jk} with probability $1 - 2\delta m$, by means of Lemma 3.4. Furthermore, employing

Lemma 3.5, we have that in each point p^{+i} either a_j or a_k is a follower's best response. As a result, by employing a union bound, we can compute $m-1$ linearly independent points lying on the separating hyperplane H_{jk} with probability at least $1 - 3\delta(m+n)^2 - 2\delta m$.

To conclude the proof, we observe that the binary search to compute p° is consistently performed between a point sampled according to Algorithm 3.4, whose bit-complexity is bounded by $\mathcal{O}(Lm^3 + \log(1/\delta))$ as a consequence of Lemma 3.8, and either another interior point or a vertex. Therefore, since the bit-complexity of each vertex can be bounded by $O(m^2L)$ by means of Lemma A.2, the number of rounds required to compute p° by means of Algorithm 3.3 is $\mathcal{O}(m^4L + m\log(1/\delta))$ as prescribed by Lemma 3.7.

Furthermore, the bit-complexity of p° is bounded by $\mathcal{O}(m^4L+m\log(1/\delta))$. As a result, the bit-complexity of the parameter α is equal to $\mathcal{O}(m^5L + m^2\log(1/\delta))$. Thus, the bit-complexity of each point $p^{+i} = p^\circ + \alpha(p - p^\circ)$ with $p \in H_i \cap \Delta_m$ is bounded by $\mathcal{O}(m^5L + m^2\log(1/\delta))$. Consequently, each of the final $m-1$ binary searches performed to compute the $m-1$ points on the separating hyperplane requires at most $\mathcal{O}(m^6L + m^3\log(1/\delta))$ samples and returns a point on the hyperplane with bit-complexity bounded by $\mathcal{O}(m^6L + m^3\log(1/\delta))$. Therefore, Algorithm 3.2 requires $\mathcal{O}(m^7L + m^4\log(1/\delta))$ samples by accounting for all the $m-1$ binary search performed. □

The number of samples can be bounded by observing that binary search is always performed between points with bit-complexity of the order of $\mathcal{O}(m^5L + m^2\log(1/\delta))$, and that $\mathcal{O}(m)$ binary searches are performed. This is ensured by Lemmas 3.7 and 3.8, and the fact that the bit-complexity of a vertex is $\mathcal{O}(m^2L)$.

3.2.3 `Binary-Search`

Algorithm 3.3 `Binary-Search`

Require: a_j, p^1, p^2 of bit-complexity $\leq B$ ▷ Algorithm 3.2

$\lambda_1 \leftarrow 0$; $\lambda_2 \leftarrow 1$
$\beta \leftarrow 2^{-6m(5B+8L)}$
while $|\lambda_2 - \lambda_1| \geq \beta$ **do**
  $\lambda \leftarrow (\lambda_1 + \lambda_2)/2$; $p^\circ \leftarrow p^1 + \lambda(p^2 - p^1)$
  if `Oracle`$(p^\circ) = a_j$ **then**
    $\lambda_1 \leftarrow \lambda$ ▷ New point with a_j as best response
  else
    $\lambda_2 \leftarrow \lambda$ ▷ New point with best response $\neq a_j$
  end if
end while
$\lambda \leftarrow$ `Stern-Brocot-Tree`$(\lambda_1, \lambda_2, 3m(5B + 8L))$
$p^\circ \leftarrow \lambda p^1 + (1 - \lambda)p^2$

The pseudocode of `Binary-Search` is provided in Algorithm 3.3. The binary search is performed on the segment with extreme points p^1, p^2, with `Oracle`$(p^1) = a_j$ and `Oracle`$(p^2) \neq a_j$. At each iteration of the binary search, the algorithm

queries the middle point of the line segment. If the best response played by the follower is a_j, then the algorithm advances one extreme of the line segment, otherwise it moves the other extreme. The binary search stops when the line segment is sufficiently small, so that a point on a new separating hyperplane can be computed.

The number of binary search steps needed to find a point on an hyperplane depends on the bit-complexity B of p^1, p^2, and the bit-complexity L of follower's payoffs. Indeed, in an interval with a suitably-defined length there is a single rational number with bit-complexity $\mathcal{O}(m(B+L))$, and this is the point on the separating hyperplane. Such a point can be efficiently computed by binary search on the Stern-Brocot tree. See [Forišek, 2007] for an efficient implementation.

Lemma 3.7. *Let $p^1, p^2 \in \Delta_m$ be two points such that (i) $p^1 \in \text{int}(\mathcal{P}_j)$ and $p^2 \notin \mathcal{P}_j$ with $\text{vol}(\mathcal{P}_j) > 0$ and (ii) they have bit-complexity bounded by B. Then, Algorithm 3.3 finds a point $p^\circ \in \text{int}(\Delta_m)$ lying on a hyperplane H_{jk} for some $a_k \in A_f$. Furthermore, the algorithm requires $\mathcal{O}(m(B+L))$ samples, and the point p° has bit-complexity bounded by $\mathcal{O}(m(B+L))$.*

Proof. Let p° be the point lying on the line segment connecting p^1 to p^2 that also lies in $\partial\mathcal{P}_j$. This point is unique by the convexity of $\mathcal{P}_j$ and by the assumption that the line segment connecting p^1 to p^2 is not contained in $\partial\mathcal{P}_j$.

We notice that the point p° lies on a facet of $\mathcal{P}_j$ contained in some separating hyperplane H_{jk}, but not in any boundary hyperplane H_i with $i \in [m]$. Indeed, since $p^1 \in \text{int}(\mathcal{P}_j) \subseteq \text{int}(\Delta_m)$ and $p^2 \notin \mathcal{P}_j$, the point p° cannot coincide with p^2, as p° also belongs to $\mathcal{P}_j$. Furthermore, there exists a parameter $\lambda^\circ \in [0,1]$ such that $p^\circ = p^1 + \lambda^\circ(p^2 - p^1)$, and $\sum_{i\in[m]} p_i^\circ\big(u_f(a_i,a_j) - u_f(a_i,a_k)\big) = 0$. By combining the two equations we obtain

$$\lambda^\circ = \frac{\left|\sum_{i=1}^m p_i^1\big(u_f(a_i,a_j) - u_f(a_i,a_k)\big)\right|}{\left|\sum_{i=1}^m (p_i^2 - p_i^1)\big(u_f(a_i,a_j) - u_f(a_i,a_k)\big)\right|}.$$

We observe that the numerator of λ° is a sum of m terms, each with bit complexity at most $B+4L$, so by Lemma A.1 its bit complexity is bounded by $3m(B+4L)$. Analogously, the denominator has bit complexity at most $3m(4B+4L)$. Hence λ° has bit complexity at most $3m(5B+8L)$.

We prove that $\lambda' = \lambda^\circ$ and thus $p' = p^\circ$. We observe that Algorithm 3.3 maintains a line segment whose endpoints correspond to different best responses of the follower, so the final interval $[\lambda_1, \lambda_2]$ contains λ°. By Lemma A.1, any two distinct rational numbers with bit complexity at most $N^\star := 3m(5B+8L)$ differ by at least β (defined in Algorithm 3.3). Since Algorithm 3.3 stops once $|\lambda_1 - \lambda_2| \leq \beta$, there is at most one rational of bit complexity smaller or equal to $N^\star$ in $[\lambda_1, \lambda_2]$. Because λ° has bit complexity at most $N^\star$ and lies in this interval, and Algorithm 3.3 returns some $\lambda' \in [\lambda_1, \lambda_2]$ with bit complexity at most $N^\star$, we conclude that $\lambda' = \lambda^\circ$.

The bit complexity of the point p° is bounded by $\mathcal{O}(m(B+L))$. Indeed, $p^\circ = p^1 + \lambda^\circ(p^2 - p^1)$ is the sum of two rational numbers with bit complexity bounded by $3m(5B+8L) + B$, which yields a bound of $24m(3B+4L)$. Lastly,

the number of rounds required by Algorithm 3.3 to terminate is $\mathcal{O}(\log_2(1/\epsilon))$, and thus $\mathcal{O}(m(B+L))$. □

3.2.4 `Sample-Int`

The pseudocode of `Sample-Int` is in Algorithm 3.4. It takes as input a polytope $\mathcal{P} \subseteq \Delta_m : \text{vol}_{m-1}(\mathcal{P}) > 0$ or a facet $\mathcal{P} := H_i \cap \Delta_m$ and a probability $\delta \in (0,1)$, and it returns a point from $\text{int}(\mathcal{P})$. First, Algorithm 3.4 computes $p^\diamond \in \text{int}(\mathcal{P})$ by taking

Algorithm 3.4 `Sample-Int`

Require: $\mathcal{P} \subseteq \Delta_m : \text{vol}_{m-1}(\mathcal{P}) > 0 \vee \mathcal{P} := H_i \cap \Delta_m, \delta$

1: $\begin{cases} d \leftarrow m & \triangleright \text{If } \mathcal{P} \subseteq \Delta_m : \text{vol}_{m-1}(\mathcal{P}) > 0 \\ d \leftarrow m-1 & \triangleright \text{If } \mathcal{P} = H_i \cap \Delta_m \end{cases}$

2: $\mathcal{V} \leftarrow d$ linearly-independent vertexes of $\mathcal{P}$

3: $p^\diamond \leftarrow \frac{1}{d}\sum_{v \in \mathcal{V}} v$

4: $\rho \leftarrow \left(d^3 2^{9d^3 L + 4dL}\right)^{-1}$; $M \leftarrow \lceil \sqrt{d}/\delta \rceil$

5: $x \sim \text{Uniform}(\{-1, -\frac{M-1}{M}, \dots, 0, \dots, \frac{M-1}{M}, 1\}^{d-1})$

6: $l \leftarrow 1$; $k \leftarrow 1$

7: **while** $l \leq d-1$ **do**

8: **if** $k \neq i$ **then** $p_k \leftarrow p^\diamond_k + \rho x_l$; $l \leftarrow l+1$

9: **else** $p_k \leftarrow 0$;

10: $k \leftarrow k+1$

11: **end while**

12: **if** $i = m$ **then** $p_{m-1} \leftarrow 1 - \sum_{l=1}^{m-2} p_l$

13: **else** $p_m \leftarrow 1 - \sum_{l=1}^{m-1} p_l$

the average of d linearly-independent vertexes of $\mathcal{P}$ (Line 3), where either $d = m$ or $d = m-1$, depending on the received input. Then, the algorithm samples a point $x \in \mathbb{R}^{d-1}$ from a discrete uniform distribution with support on points belonging to an hypercube (Line 5). In particular, the support is an equally-spaced grid in the hypercube, with step $1/M$ (Line 4). Then, the algorithm adds each component x_l of x to a component $p^\diamond_k$ of $p^\diamond$, by scaling it by a factor ρ (Line 4). The last component of p is set so that the p_i sum to 1. Notice that the choice of ρ is done to guarantee that the p produced by Algorithm 3.4 belongs to $\text{int}(\mathcal{P})$. The main property of Algorithm 3.4 is that, given any affine space $H \subseteq \mathbb{R}^m$ (with $\mathcal{P} \not\subseteq H$), the probability that the sampled point belongs to H is at most δ. Notice that, when existing algorithms [Peng et al., 2019, Letchford et al., 2009] randomly sample a point, they consider the probability of such an event to be zero, since they use strategies with arbitrarily large bit-complexity. The property is made formal in Lemma 3.8, which at a very high level follows by bounding the volume of the intersection of the hypercube with a linear space of dimension at most $m-1$ [Ball, 1986, Ivanov, 2021], and by showing that the bit-complexity of all elements involved is suitably bounded.

Lemma 3.8. *Let $\mathcal{P} \subseteq \Delta_m$ be a polytope with $\text{vol}_{m-1}(\mathcal{P}) > 0$, defined either by separating or boundary hyperplanes, or as a facet $\mathcal{P} := H_i \cap \Delta_m$. Algorithm 3.4 computes a point $p \in \Delta_m$ satisfying: (1) For every affine subspace $H \subset \mathbb{R}^m$ such that $\dim(H) \leq m-1$*

and $\mathcal{P} \nsubseteq H$, it holds that $\mathbb{P}(p \in H) \leq \delta$; (2) The point p lies in $\operatorname{int}(\mathcal{P})$; (3) The bit complexity of p is at most $40m^3L + 2\log_2(1/\delta)$.

Proof. In the following, for the sake of the presentation, we provide the proof of the lemma when the polytope $\mathcal{P}$ is such that $\operatorname{vol}_{m-1}(\mathcal{P}) > 0$, the case when $\mathcal{P} = H_i \cap \Delta_m$ can be proved with the same analysis.

1. Let
$$H \coloneqq \{x \in \mathbb{R}^m \mid \sum_{i=1}^{m} \alpha_i x_i = \beta\}$$
be a hyperplane in $\mathbb{R}^m$. We show that the probability that the commitment p returned by Algorithm 3.4 belongs to H equals the probability that a point x sampled uniformly from $\Xi \coloneqq \{-1, -\frac{M-1}{M}, \dots, 0, \dots, \frac{M-1}{M}, 1\}^{m-1}$ lies in a suitably defined affine space H'.
$$\begin{aligned}
\mathbb{P}(p \in H) &= \mathbb{P}\left(\sum_{i=1}^{m-1} \alpha_i p_i + \alpha_m \left(1 - \sum_{i=1}^{m-1} p_i\right) = \beta\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, p_i = \beta - \alpha_m\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, (p_i^\diamond + \rho x_i) = \beta - \alpha_m\right) \\
&= \mathbb{P}\left(\rho \sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, x_i = \beta - \alpha_m - \sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, p_i^\diamond\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, x_i = \frac{1}{\rho}\left(\beta - \alpha_m - \sum_{i=1}^{m-1} (\alpha_i - \alpha_m) p_i^\diamond\right)\right) \\
&= \mathbb{P}\left(x \in H'\right),
\end{aligned}$$
where we let
$$H' = \left\{x \in \mathbb{R}^{m-1} \mid \sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, x_i = \frac{1}{\rho}\left(\beta - \alpha_m - \sum_{i=1}^{m-1} (\alpha_i - \alpha_m)\, p_i^\diamond\right)\right\}.$$
Furthermore, we observe that a discrete uniform probability over the set Ξ can be written as follows:
$$\mathbb{P}_{x \sim \mathcal{U}(\Xi)}(x = \bar{x}) = \frac{1}{\operatorname{vol}_{m-1}(Q)} \int_Q \mathbb{1}\left\{\|s - \bar{x}\|_\infty \leq \frac{1}{2M}\right\} ds$$
for each $\bar{x} \in \Xi$, where
$$Q \coloneqq \{x \in \mathbb{R}^{m-1} \mid |x_i| \leq 1 + 1/2M \ \forall i \in [m-1]\}$$

is a hypercube in dimension $m-1$. Then, the following holds:

$$\begin{aligned}
\mathbb{P}_{x\sim\mathcal{U}(\Xi)}\left(x\in H'\right) &= \sum_{\bar{x}\,\in\,\Xi}\mathbb{1}\left\{\bar{x}\in H'\right\}\mathbb{P}_{x\sim\mathcal{U}(\Xi)}\left(x=\bar{x}\right)\\
&= \frac{1}{\mathrm{vol}_{m-1}(Q)}\sum_{\bar{x}\,\in\,\Xi}\mathbb{1}\left\{\bar{x}\in H'\right\}\int_Q \mathbb{1}\left\{\|s-\bar{x}\|_\infty \le \frac{1}{2M}\right\}ds\\
&= \frac{1}{\mathrm{vol}_{m-1}(Q)}\sum_{\bar{x}\,\in\,\Xi}\int_Q \mathbb{1}\left\{\|s-\bar{x}\|_\infty \le \frac{1}{2M}, \bar{x}\in H'\right\}ds\\
&= \frac{1}{\mathrm{vol}_{m-1}(Q)}\int_Q \mathbb{1}\left\{\bar{x}\in H'\cap\Xi \mid \|s-\bar{x}\|_\infty \le \frac{1}{2M}\right\}ds\\
&\le \frac{1}{\mathrm{vol}_{m-1}(Q)}\int_Q \mathbb{1}\left\{d_\infty(s,H') \le \frac{1}{2M}\right\}ds\\
&\le \frac{1}{\mathrm{vol}_{m-1}(Q)}\int_Q \mathbb{1}\left\{d_2(s,H') \le \frac{\sqrt{m}}{2M}\right\}ds
\end{aligned}$$

Thus, we can bound the probability that the commitment sampled according to Algorithm 3.4 belongs to the separating hyperplane H as follows:

$$\begin{aligned}
\mathbb{P}(p\in H) = \mathbb{P}_{x\sim\mathcal{U}(\Xi)}\left(x\in H'\right) &\le 2\frac{\displaystyle\int_{Q\cap H'}\frac{\sqrt{m}}{2M}dA}{\mathrm{vol}_{m-1}(Q)}\\
&= \frac{\sqrt{m}}{M}\frac{\displaystyle\int_{Q\cap H'}dA}{(2+1/M)^{m-1}}\\
&\le \frac{\sqrt{2m}}{M}\frac{(2+1/M)^{m-2}}{(2+1/M)^{m-1}}\\
&= \frac{\sqrt{m}}{\sqrt{2}M} \le \delta.
\end{aligned}$$

Here dA denotes the Lebesgue measure restricted to the affine subspace H', with dimension $m-2$. We notice the second inequality above holds by employing the results by Ball [1986] and Ivanov [2021] and the last inequality follows since $M = \lceil\sqrt{m}/\delta\rceil$, showing that $\mathbb{P}(p\in H)\le\delta$.

Furthermore, we notice that for each affine space $\widetilde{H}$ of dimension less than $m-1$, there exists an affine space H of dimension $m-1$ such that $\widetilde{H}\subseteq H$ and $\mathcal{P}\not\subseteq H$, and thus

$$\mathbb{P}(p\in\widetilde{H}) \le \mathbb{P}(p\in H) \le \delta.$$

2. We show that the point p sampled by Algorithm 3.4 lies in the interior of the polytope $\mathcal{P}$. For this, let v^h denote the h-th vertex of the set $\mathcal{V}$, so that $\mathcal{V} = \{v^1,\ldots,v^m\}$ with $V = |\mathcal{V}|$. Since $p^\diamond \in \mathrm{int}(\mathcal{P})$, the $\|\cdot\|_2$-distance

between $p^\diamond$ and any separating hyperplane H_{jk} defining the boundary of $\mathcal{P}$ admits the following lower bound:

$$\begin{aligned} d_2(p^\diamond, H_{jk}) &= \left| \frac{\sum_{i=1}^{m} p_i^\diamond \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big)}{\sqrt{\sum_{i=1}^{m} \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big)^2}} \right| \\ &= \left| \frac{\sum_{i=1}^{m} \sum_{h=1}^{m} \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big) v_i^h}{m \sqrt{\sum_{l=1}^{m} \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big)^2}} \right| \\ &\geq \left(m^2 2^{9m^3 L + 2mL} \right)^{-1} =: r. \end{aligned} \tag{3.1}$$

To prove this, we lower bound the numerator and upper bound the denominator in Equation (3.1). In particular, the denominator is at most m^2, since $|u_f(a_i, a_j) - u_f(a_i, a_k)| \leq 1$ for every $i \in [m]$. To lower bound the numerator, we define:

$$\frac{\beta_i}{\gamma_i} := u_f(a_i, a_j) - u_f(a_i, a_k) \quad \forall i \in [m] \quad \text{and} \quad v_i^h := \frac{\mu_i^h}{\nu_h} \quad \forall v^h \in \mathcal{V},$$

where we let β_i and γ_i be integer numbers for each $i \in [m]$, while ν_h and μ_i^h are natural numbers for each $i \in [m]$ and $h \in [V]$.

As a result, we have:

$$\begin{aligned} &\left| \sum_{i=1}^{m} \sum_{h=1}^{m} \Big(u_f(a_i, a_j) - u_f(a_i, a_k) \Big) v_i^h \right| \\ &\qquad = \left| \sum_{i=1}^{m} \sum_{h=1}^{m} \frac{\beta_i \mu_i^h}{\gamma_i \nu_h} \right| \\ &\qquad = \left| \frac{\sum_{i=1}^{m} \sum_{h=1}^{m} \beta_i \mu_i^h \left(\prod_{i' \neq i} \gamma_{i'} \prod_{h' \neq h} \nu_{h'} \right)}{\prod_{i=1}^{m} \gamma_i \prod_{h=1}^{m} \nu_h} \right| \\ &\qquad \geq \left(\prod_{i=1}^{m} \gamma_i \prod_{h=1}^{m} \nu_h \right)^{-1} \\ &\qquad \geq 2^{-(9m^3 L + 4mL)}. \end{aligned} \tag{3.2}$$

We observe that the first inequality holds because the numerator in Equation (3.2) can be lower bounded by one, since $p^\diamond \in \text{int}(\mathcal{P})$. On the other hand, the denominator in Equation (3.2) can be upper bounded by noting that the bit-complexity of γ_i is at most $4L$ for each $i \in [m]$, as ensured by Lemma A.1. Moreover, for every $h \in [m]$, the bit-complexity of ν_h is at most $9m^2L$ by Lemma A.2. By a similar argument, we can prove that the $\|\cdot\|_2-$distance between $p^\diamond$ and any boundary hyperplane is lower bounded by r.

As a final step, we bound the $\|\cdot\|_2-$distance between $p^\diamond$ and p. Formally:

$$\begin{aligned} d_2(p^\diamond, p) &= \sqrt{\sum_{i\in[m-1]} \left(p_i^\diamond + \rho x_i - p_i^\diamond\right)^2 + \left(p_m^\diamond - 1 + \sum_{i\in[m-1]} p_i^\diamond + \rho x_i\right)^2} \\ &\leq \sqrt{\sum_{i\in[m-1]} \left(p_i^\diamond + \rho x_i - p_i^\diamond\right)^2 + \left(\sum_{i\in[m]} p_i^\diamond + \sum_{i\in[m-1]} \rho x_i - 1\right)^2} \\ &\leq \sqrt{\sum_{i\in[m-1]} \left(\rho x_i\right)^2 + \left(\sum_{i\in[m-1]} \rho x_i\right)^2} \\ &\leq \sqrt{(m-1)\rho^2 + (m-1)^2\rho^2} = \rho\sqrt{(m-1)m} < \rho m \end{aligned}$$

As a result, if $\rho = r/m$, the distance $d_2(p^\diamond, p)$ is strictly smaller than r, which shows that the commitment p lies in $\text{int}(\mathcal{P})$.

3. We also show that the bit complexity of p is bounded by $40m^3L + 2\log_2(1/\delta)$. To do so, we observe that the denominator of each rational number x_i with $i \in [m-1]$ is equal to $M = \lceil\sqrt{m}/\delta\rceil$. Furthermore, each component of $p^\diamond$ can be expressed as follows:

$$p_i^\diamond = \frac{1}{m}\sum_{h=1}^{m} v_i^h = \frac{1}{m}\sum_{h=1}^{m} \frac{\mu_i^h}{\nu_h} = \frac{\sum_{i=1}^{m} \mu_i^h \left(\prod_{h\neq h'} \nu_{h'}\right)}{m\prod_{h=1}^{m} \nu_h},$$

for each $i \in [m]$.

Consequently, the denominator of each rational component $p_i = p_i^\diamond + \rho x_i$ with $i \in [m-1]$ is equal to $D := m\prod_{h=1}^{m} \nu_h D_\rho M$, where D_ρ represents the denominator defining the rational number ρ. Additionally, we observe that the last component p_m can also be expressed as a rational number with the same denominator D.

Finally, since $p_i \in [0,1]$ for every $i \in [m]$, its bit complexity is bounded by $2B_D$, where the bit complexity of the denominator D can be upper

bounded as follows:

$$
\begin{aligned}
B_D &= \left\lceil \log_2 \left(m \prod_{h=1}^{m} \nu_h D_\rho M \right) \right\rceil \\
&= \left\lceil \log_2(m) + \sum_{h=1}^{m} \log_2(\nu_h) + \log_2(m 2^{9m^3 L + 4mL}) + \log_2(M) \right\rceil \\
&\leq 1 + \log_2(m) + \sum_{h=1}^{m} \log_2(2^{9m^2 L}) + \log_2(m 2^{9m^3 L + 4mL}) + \log_2(\sqrt{m}/\delta + 1) \\
&\leq 20 m^3 L + \log_2(1/\delta).
\end{aligned}
$$

Thus, the bit complexity of p is bounded by $40m^3 L + 2\log(1/\delta)$, concluding the proof. □

3.3 Related works

The two works closely related to ours are the ones by Letchford et al. [2009] and Peng et al. [2019]. In addition to these two works, the problem of learning optimal strategies in Stackelberg games has received significant attention from the scientific community. While we examined a setting with a finite number of actions and where the leader can commit to mixed strategies, Fiez et al. [2020] study games with continuous action spaces where the leader commits to a single action. They present an algorithm based on a gradient-descent approach. Bai et al. [2021] propose a model where both the leader and the follower learn through repeated interactions, observing only noisy samples of their rewards. Lauffer et al. [2022] study Stackelberg games characterized by a random state that influences the leader's utility and the set of available actions.

Our work is also related to the problem of learning optimal strategies in Stackelberg security games. In these settings a defender needs to allocate a set of resources to protect some targets against one or multiple attackers. Blum et al. [2014] provide an algorithm to learn an approximately optimal strategies in polynomial time in the number of targets. Balcan et al. [2015] consider instead a sequence of attackers of different types. In particular, they focus on the case in which an adversary chooses the attacker's types from a known set and they design a no-regret algorithm. Haghtalab et al. [2016] focuses instead on security games where the attacker is not completely rational. Thanks to this assumption, they develop an algorithm that can be executed offline on an existing dataset. Finally, the algorithm proposed by Peng et al. [2019] for general Stackelberg games can be modified to tackle directly Stackelberg security games in their natural representation.

Part II

Learning in Bayesian Persuasion

CHAPTER 4

Learning to persuade a single receiver

In this chapter, we focus—for the first time to the best of our knowledge—on online Bayesian persuasion settings in which an informed sender interacts with a receiver over T rounds, without knowing the receiver's utility function or the prior distribution over the set of states of nature. At each round the sender commits to a signaling scheme, observes the realized state of nature, and sends the corresponding signal to the receiver; afterwards, the only feedback they receive is the best-response action played by the receiver. In this setting, the sender's goal is to design a no-regret algorithm with respect to an optimal signaling scheme.

Our main result is an algorithm that achieves $\widetilde{\mathcal{O}}(\sqrt{T})$ regret whenever either the number of states d or the number of receiver actions n is fixed. Our algorithm implements a sophisticated explore-then-commit scheme, where exploration is performed in a suitable space of signaling schemes so as to learn the receiver's best responses exactly. This is crucial to attaining tight regret guarantees, and it is made possible by a non-standard representation of signaling schemes, which allows us to overcome the challenge posed by the lack of knowledge of both the prior and the receiver's utilities.

We also establish matching upper and lower bounds on the sample complexity of learning an optimal signaling scheme, where the goal is to identify, with high probability, an approximately optimal scheme using as few sender-receiver rounds as possible without prior knowledge of the receiver's utility or the distribution over states of nature. Finally, we extend these results to the case where the prior is known but the receiver's utility is not, and we derive matching upper and lower bounds.

4.1 Online learning in Bayesian persuasion

We study settings in which the sender *repeatedly* interacts with the receiver over multiple rounds, with each round involving a one-shot Bayesian persuasion interaction. We assume that the sender has *no* knowledge about both the prior μ and receiver's utility u, and that the only feedback they get after each round is the best-response action played by the receiver.

At each round $t \in [T]$, the sender commits to $\phi_t : \Theta \to \Delta_{\mathcal{S}}$ and observes a state of nature $\theta^t \sim \mu$. Then, they draw a signal $s^t \sim \phi_{t,\theta^t}$ and send it to the receiver, who plays a best-response action $a^t \coloneqq a^{\phi_t}(s^t)$. Finally, the sender gets payoff $u_t^{\mathrm{s}} \coloneqq u_{\theta^t}^{\mathrm{s}}(a^t)$ and observes a *feedback* consisting in the action a^t played by the receiver. The goal of the sender is to learn how to maximize their expected utility while repeatedly interacting with the receiver. When the sender commits to a sequence $\{\phi_t\}_{t\in[T]}$ of signaling schemes, their performance over the T rounds is measured by means of the following notion of *cumulative* (*Stackelberg*) *regret*:

$$R_T(\{\phi_t\}_{t\in[T]}) \coloneqq T \cdot \mathrm{OPT} - \mathbb{E}\left[\sum_{t=1}^{T} u^{\mathrm{s}}(\phi_t)\right],$$

where the expectation is over the algorithm's randomness and the environment's stochasticity. In the following, for ease of notation, we omit the dependency on $\{\phi_t\}_{t\in[T]}$ from the cumulative regret, by simply writing R_T. Then, our goal is to design *no-regret* learning algorithms, which prescribe a sequence of signaling schemes ϕ_t that results in the regret R_T growing sublinearly in T, *i.e.*, $R_T = o(T)$.

Similarly to Chapter 3, we assume that all numbers manipulated by our algorithms are rational. We further assume that rational numbers are represented as fractions, specified by two integers encoding their numerator and denominator. For a rational number $q \in \mathbb{Q}$ represented as b/c with $b, c \in \mathbb{Z}$, we define its *bit-complexity* as the number of bits needed to store it, namely $B_q \coloneqq B_b + B_c$, where B_b (resp. B_c) is the number of bits required to represent the numerator (resp. denominator). For simplicity, and with a slight abuse of terminology, given a vector in $\mathbb{Q}^D$ of D rational entries, we let its bit-complexity be the maximum bit-complexity among its components. Finally, we assume that the bit-complexity of both the receiver's utility and the prior distribution is bounded. Specifically, we denote by B_μ the bit-complexity of the prior μ, and by B_u an upper bound on the bit-complexity of each d-dimensional vector $u_\theta(a)$ with $\theta \in \Theta$. Moreover, we let $B \coloneqq B_\mu + B_u$.

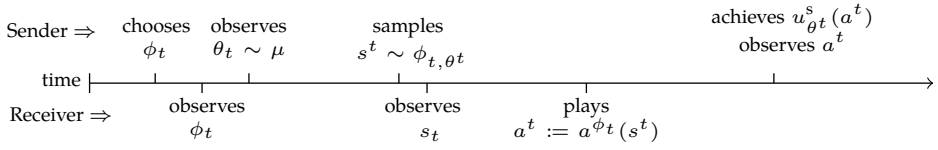

Figure 4.1: *Interaction between the sender and the receiver at each round $t \in [T]$.*

4.2 Warm-up: a single signal is all you need

In order to design our learning algorithm in Section 4.3, we exploit a non-standard representation of signaling schemes, which we introduce in this section. Adopting such a representation is fundamental to be able to learn receiver's best responses *without* any knowledge of both the prior μ and receiver's utility function u. The crucial observation that motivates its adoption is that receiver's best responses $a^{\phi}(s)$ only depend on the components of ϕ associated with $s \in \mathcal{S}$, namely $\phi_\theta(s)$ for $\theta \in \Theta$. Thus, in order to learn them, it is sufficient to learn how $a^{\phi}(s)$ varies as a function of such components. The signaling scheme representation introduced in this section revolves around the concept of *slice*.

Definition 4.1 (Slice)**.** *Given a signaling scheme $\phi : \Theta \to \Delta_{\mathcal{S}}$, the* slice of ϕ with respect to signal $s \in \mathcal{S}$ *is the d-dimensional vector $x \in [0,1]^d$ with components $x_\theta := \phi_\theta(s)$ for $\theta \in \Theta$.*

In the following, we denote by $\mathcal{X}^{\square} := [0,1]^d$ the set of *all* the possible slices of signaling schemes. Moreover, we let $\mathcal{X}^{\triangle}$ be the set of *normalized* slices, which is simply obtained by restricting slices to lie in the $(d-1)$-dimensional simplex. Thus, it holds that

$$\mathcal{X}^{\triangle} := \left\{ x \in [0,1]^d \mid \sum_{\theta \in \Theta} x_\theta = 1 \right\}.$$

Next, we show that receiver's actions induce particular coverings of the sets $\mathcal{X}^{\square}$ and $\mathcal{X}^{\triangle}$, which also depend on both the prior μ and receiver's utility u. First, we introduce $\mathcal{H}_{ij} \subseteq \mathbb{R}^d$ to denote the halfspace of slices under which action $a_i \in A$ is (weakly) better than action $a_j \in A$ for the receiver.

$$\mathcal{H}_{ij} := \left\{ x \in \mathbb{R}^d \mid \sum_{\theta \in \Theta} x_\theta \mu_\theta \big(u_\theta(a_i) - u_\theta(a_j)\big) \geq 0 \right\}.$$

Moreover, we denote by $H_{ij} := \partial \mathcal{H}_{ij}$ the hyperplane constituting the boundary of the halfspace $\mathcal{H}_{ij}$, which we call the *separating hyperplane* between actions a_i and a_j.[1] Then, we introduce the polytopes $\mathcal{X}^{\square}(a_i) \subseteq \mathcal{X}^{\square}$ and $\mathcal{X}^{\triangle}(a_i) \subseteq \mathcal{X}^{\triangle}$:

$$\mathcal{X}^{\square}(a_i) := \mathcal{X}^{\square} \cap \left(\bigcap_{a_j \in \mathcal{A} : a_i \neq a_j} \mathcal{H}_{ij} \right) \text{ and } \mathcal{X}^{\triangle}(a_i) := \mathcal{X}^{\triangle} \cap \left(\bigcap_{a_j \in \mathcal{A} : a_i \neq a_j} \mathcal{H}_{ij} \right),$$

for every $a_i \in \mathcal{A}$. Clearly, the sets $\mathcal{X}^{\square}(a_i)$, respectively $\mathcal{X}^{\triangle}(a_i)$, define a cover of $\mathcal{X}^{\square}$, respectively $\mathcal{X}^{\triangle}$. Intuitively, the set $\mathcal{X}^{\square}(a_i)$ encompasses all the slices under which action a_i is a best response for the receiver. The set $\mathcal{X}^{\triangle}(a_i)$ has the same interpretation, but for normalized slices. Specifically, if $x \in \mathcal{X}^{\square}(a_i)$ is a slice of ϕ with respect to $s \in \mathcal{S}$, then $a_i \in \mathcal{A}^{\phi}(s)$. Notice that a slice $x \in \mathcal{X}^{\square}$ may belong to more than one $\mathcal{X}^{\square}(a_i)$, when $|\mathcal{A}^{\phi}(s)| > 1$ for signaling schemes ϕ having x as slice with respect to $s \in \mathcal{S}$. To denote the best-response action actually played by the receiver under a slice $x \in \mathcal{X}^{\square}$, we introduce the symbol $a(x)$, where $a(x) := a^{\phi}(s)$ for any ϕ having x as slice with respect to $s \in \mathcal{S}$.

[1] We let $\partial \mathcal{H}$ be the boundary hyperplane of halfspace $\mathcal{H} \subseteq \mathbb{R}^d$. Notice that H_{ij} and H_{ji} actually refer to the same hyperplane. In this chapter, we use both names for ease of presentation.

A crucial fact exploited by the learning algorithm developed in Section 4.3 is that knowing the separating hyperplanes H_{ij} defining the polytopes $\mathcal{X}^{\triangle}(a_i)$ of normalized slices is *sufficient* to determine an optimal signaling scheme. In particular, as we show in the following Section 4.3 (see Lemma 4.6 in particular), an optimal signaling scheme can be computed by suitably combining normalized slices representing vertices of polytopes $\mathcal{X}^{\triangle}(a_i)$. Intuitively, this follows from the fact that each $\mathcal{X}^{\square}(a_i)$ is obtained by intersecting a "cone-shaped" space with the hypercube $[0,1]^d$, since all the separating hyperplanes pass through the origin in $\mathbb{R}^d$. Thus, if one takes any slice $x \in \mathcal{X}^{\square}(a_i)$ and "scales" it by some scalar value α, then the resulting slice is still in $\mathcal{X}^{\square}(a_i)$. Figure 4.2 depicts an example of the polytopes $\mathcal{X}^{\square}(a_i)$ and $\mathcal{X}^{\triangle}(a_i)$ to help the reader to grasp more intuition on them.

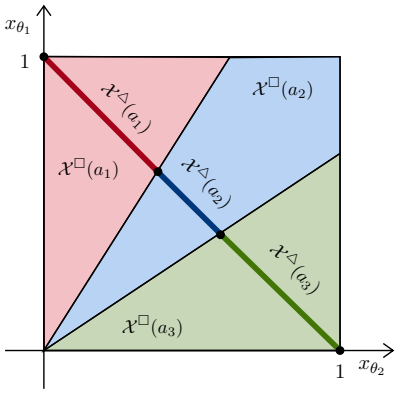

Figure 4.2: *Representation of sets $\mathcal{X}^{\square}(a_i)$ and $\mathcal{X}^{\triangle}(a_i)$ for an instance with $d = 2$ states of nature and $n = 3$ receivers' actions.*

We conclude the section with some remarks that help to better clarify why we need to work with signaling scheme slices in order to design our learning algorithm in Section 4.3.

Why we need slices for learning. The coefficients of separating hyperplanes H_{ij} are products between prior probabilities μ_θ and receiver's utility differences $u_\theta(a_i) - u_\theta(a_j)$. In order to design a no-regret learning algorithm, it is fundamental that such coefficients are learned exactly, since even an arbitrarily small approximation error may result in "missing" some receiver's best responses, and this may potentially lead to a large loss in sender's expected utility (and, in its turn, to large regret). As a result, any naïve approach that learns the prior and receiver's payoffs separately is deemed to fail, as it would inevitably result in approximate separating hyperplanes being learned. Operating in the space of signaling scheme slices crucially allows us to learn separating hyperplanes exactly. As we show in Section 4.3, it makes it possible to directly learn the coefficients of separating hyperplanes without splitting them into products of prior probabilities and receiver's utility differences.

Why we need normalized slices. One may wonder why we cannot work with (unnormalized) slices in $\mathcal{X}^{\square}$, rather than with normalized ones in $\mathcal{X}^{\triangle}$. Indeed, as we show in Section 4.3, our procedure to learn separating hyperplanes crucially relies on the fact that we can restrict the search space to a subset of the $(d-1)$-dimensional simplex. This makes it possible to avoid always employing a number of rounds exponential in d, which would lead to non-tight regret guarantees.

Why *not* working with posteriors. In Bayesian persuasion, it is oftentimes useful to work in the space of posterior distributions induced by sender's signals (see, *e.g.*, [Castiglioni et al., 2020b]). These are the beliefs computed by the receiver according to Bayes rule at step (3) of the interaction. Notice that posteriors do *not* only depend on the signaling scheme ϕ and the sent signal $s \in \mathcal{S}$, but also on the prior distribution μ. Indeed, the same signaling scheme may induce different posteriors for different prior distributions. Thus, since in our setting the sender has no knowledge of μ, we cannot employ posteriors. Looking at signaling scheme slices crucially allows us to overcome the lack of knowledge of the prior. Indeed, one way of thinking of them is as "prior-free" posterior distributions.

4.3 Learning to persuade without a clue

In this section, we design our no-regret algorithm (Algorithm 4.1). We adopt a sophisticated *explore-then-commit* approach that exploits the signaling scheme representation based on slices introduced in Section 4.2. Specifically, our algorithm works by first exploring the space $\mathcal{X} \coloneqq \mathcal{X}^{\triangle}$ of normalized slices in order to learn a satisfactory subset of vertexes of the polytopes $\mathcal{X}(a_i) \coloneqq \mathcal{X}^{\triangle}(a_i)$.[2] Then, it exploits them in order to compute suitable approximately-optimal signaling schemes to be employed in the remaining rounds. Effectively implementing this approach raises considerable challenges.

The *first* challenge is that the algorithm cannot directly "query" a slice $x \in \mathcal{X}$ to know action $a(x)$, as it can only commit to fully-specified signaling schemes. Indeed, even if the algorithm commits to a signaling scheme including the selected slice $x \in \mathcal{X}$, the probability that the signal associated with x is sent depends on the (unknown) prior, as it is equal to $\sum_{\theta \in \Theta} \mu_\theta x_\theta$. This probability can be arbitrarily small. Thus, in order to observe $a(x)$, the algorithm may need to commit to the signaling scheme for an unreasonably large number of rounds. To circumvent this issue, we show that it is possible to focus on a subset $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ of normalized slices "inducible" with at least a suitably-defined probability $\epsilon \in (0, 1)$. Such a set $\mathcal{X}_\epsilon$ is built by the algorithm in its first phase.

The *second* challenge that we face is learning the vertices of polytopes $\mathcal{X}_\epsilon(a_i) \coloneqq \mathcal{X}(a_i) \cap \mathcal{X}_\epsilon$ *exactly*. This is done by means of a technically-involved procedure that learns the separating hyperplanes H_{ij} needed to identify them. This procedure is an adaptation to Bayesian persuasion settings of the algorithm proposed in Chapter 3 working in Stackelberg settings.

Finally, the *third* challenge is how to use the vertices of polytopes $\mathcal{X}_\epsilon(a_i)$ to compute suitable approximately-optimal signaling schemes to commit to after explo-

[2] In the rest of the chapter, for ease of notation, we write $\mathcal{X}$ and $\mathcal{X}(a_i)$ instead of $\mathcal{X}^{\triangle}$ and $\mathcal{X}^{\triangle}(a_i)$.

ration. We show that this can be done by solving an LP, which, provided that the set $\mathcal{X}_\epsilon$ is carefully constructed, gives signaling schemes with sender's expected utility sufficiency close to that of an optimal signaling scheme. The pseudocode of our no-regret learning algorithm is provided in Algorithm 4.1. In the pseudocode, we assume that all the sub-procedures have access to the current round counter t, all the observed states of nature θ^t, and all the feedbacks a^t, u_t^s received by the sender.[3] Moreover, in Algorithm 4.1 and its sub-procedures, we use $\widehat{\mu}_t \in \Delta_\Theta$ to

Algorithm 4.1 `Learn-to-Persuade-w/o-Clue`

Require: $T \in \mathbb{N}$

$\delta \leftarrow 1/T$
$\zeta \leftarrow 1/T$
$\epsilon \leftarrow \left\lceil \frac{\sqrt{Bn}d^4}{\sqrt{T}} \right\rceil$
$T_1 \leftarrow \left\lceil \frac{12}{\epsilon} \log\left(\frac{2d}{\delta}\right) \right\rceil$
$t \leftarrow 1$
$\mathcal{X}_\epsilon \leftarrow$ `Build-Search-Space`(T_1, ϵ)
$\mathcal{Y} \leftarrow$ `Find-Slices`$(\mathcal{X}_\epsilon, \zeta)$
while $t \leq T$ **do**
    $\phi_t \leftarrow$ `Compute-Signaling`$(\mathcal{Y}, \mathcal{X}_\epsilon, \widehat{\mu}_t)$
    Commit to ϕ_t, observe θ^t, and send s^t
    Observe feedback a^t and receive u_t^s
    Compute prior estimate $\widehat{\mu}_{t+1}$
    $t \leftarrow t + 1$
end while

denote the *prior estimate* at any round $t > 1$, which is a vector with components defined as $\widehat{\mu}_{t,\theta} \coloneqq N_{t,\theta}/(t-1)$ for $\theta \in \Theta$, where $N_{t,\theta} \in \mathbb{N}$ denotes the number of times that state of nature θ is observed up to round t (excluded).

Algorithm 4.1 can be conceptually divided into three phases. In *phase 1*, the algorithm employs the `Build-Search-Space` procedure (Algorithm 4.2) to build a suitable subset $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ of "inducible" normalized slices. Then, in *phase 2*, the algorithm employs the `Find-Slices` procedure (see Algorithm 4.4) to find a $\mathcal{Y} \coloneqq (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$, where $\mathcal{V}$ is a set of vertices of the polytopes $\mathcal{X}_\epsilon(a_i)$ and $(a(x))_{x\in\mathcal{V}}$ is a tuple specifying the action $a(x)$ played by the receiver in each $x \in \mathcal{V}$. Finally, *phase 3* uses the remaining rounds to exploit the knowledge acquired in the preceding two phases. Specifically, at each t, this phase employs the `Compute-Signaling` procedure to compute an approximately-optimal signaling scheme, by using $\mathcal{Y}$, the set $\mathcal{X}_\epsilon$, and the current prior estimate $\widehat{\mu}_t$.

In the rest of this section, we describe in detail the three phases of Algorithm 4.1, bounding the regret attained by each of them.This allows us to prove the following main result about Algorithm 4.1.

Theorem 4.1. *The regret attained by Algorithm 4.1 is*

$$R_T \leq \widetilde{\mathcal{O}}\left(\binom{d+n}{d} n^{3/2} d^3 \sqrt{BT}\right).$$

[3] Notice that, in Algorithm 4.1, the sub-procedures `Build-Search-Space` and `Find-Slices` perform some rounds of interaction, and, thus, they update the current round counter t. For ease of presentation, we assume that, whenever $t > T$, their execution is immediately stopped (as well as the execution of Algorithm 4.1).

Proof. In the following, we let $\delta = \zeta = 1/T$ and $\epsilon = \lceil \sqrt{Bn} d^4 \rceil / \lceil \sqrt{T} \rceil$, as defined in Algorithm 4.1. To prove the theorem, we decompose the regret suffered in the three phases of Algorithm 4.1.

1. Phase 1. We observe that the number of rounds to execute Algorithm 4.2 is equal to $T_1 = \mathcal{O}\left(1/\epsilon \log\left(1/\delta\right) \log(d)\right)$. Thus, the cumulative regret of Phase 1 can be upper bounded as follows:

$$R_T^1 \leq \widetilde{\mathcal{O}}\left(\frac{1}{\epsilon}\log(T)\log(d)\right) \leq \widetilde{\mathcal{O}}\left(\frac{\sqrt{T}}{d^4\sqrt{nB}}\log(d)\right) \leq \widetilde{\mathcal{O}}\left(\sqrt{T}\right).$$

   This is because, at each round, the regret suffered during the execution of Algorithm 4.2 is at most one.

2. Phase 2. Thanks to Lemma 4.3, under event $\mathcal{E}_1$, with probability at least $1-\zeta$, Algorithm 4.4 terminates correctly using a number of rounds

$$\begin{aligned} T_2 &\leq \widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\zeta}\right)\left(d^7(B + B_\epsilon + B_{\widehat{\mu}}) + \binom{d+n}{d}\right)\right) \\ &\leq \widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2(T)\left(d^7(B + B_\epsilon) + \binom{d+n}{d}\right)\right), \end{aligned}$$

   where the last equality holds because $B_{\widehat{\mu}} = \mathcal{O}(\log(1/\epsilon) + \log(d) + \log(T))$. As a result, the regret suffered in Phase 2 by Algorithm 4.1 can be upper bounded as follows:

$$R_T^2 \leq \widetilde{\mathcal{O}}\left(n^{3/2}d^3\binom{d+n}{d}\sqrt{BT}\right),$$

   since, at each round, the regret suffered during the execution of Algorithm 4.4 is at most one.

3. Phase 3. Let $\tau > 0$ be the number of rounds required for Phase 1 and Phase 2 to terminate. Under the event $\mathcal{E}_1 \cap \mathcal{E}_2$, which holds with probability at least $1 - \delta - \zeta$, Lemma 4.6 guarantees that the solution returned by Algorithm 4.5 at each round $t > \tau$ is $\mathcal{O}(dn\epsilon + \nu_t)$-optimal, where we define

$$\nu_t := \left|\sum_{\theta \in \Theta} \mu_\theta - \widehat{\mu}_{t,\theta}\right|.$$

   We introduce the following event $E_t = \left\{|\mu_\theta - \widehat{\mu}_{t,\theta}| \leq \epsilon_t \quad \forall \theta \in \Theta \; \forall t > 0\right\}$, where we let $\epsilon_t > 0$ be defined as follows:

$$\epsilon_t = \sqrt{\frac{\log\left(2dT/\iota\right)}{2t}},$$

with $t > 0$. Then, by Hoeffding's inequality and a union bound we have:

$$\mathbb{P}\Big(\bigcap_{t>0} E_t\Big) \geq 1 - \iota.$$

Thus, by setting $\iota = 1/T$, the regret suffered in Phase 3 by Algorithm 4.1 can be upper bounded as follows:

$$\begin{aligned} R_T^3 &\leq \sum_{t=\tau+1}^{T} \left| \sum_{\theta \in \Theta} \mu_\theta - \widehat{\mu}_{t,\theta} \right| + \mathcal{O}\left(d\epsilon T\right) \\ &\leq d \sum_{t=1}^{T} \epsilon_t + \mathcal{O}\left(dn\epsilon T\right) \\ &\leq \widetilde{\mathcal{O}}\left(d\sqrt{T} + dn\epsilon T\right) \\ &= \widetilde{\mathcal{O}}\left(d\sqrt{T} + n^{3/2} d^5 \sqrt{BT}\right). \end{aligned}$$

As a result, by combining the regret suffered in the different phases, the regret of Algorithm 4.1 can be upper bounded as follows:

$$\begin{aligned} R_T &\leq \widetilde{\mathcal{O}}\left(n^{3/2} d^3 \binom{d+n}{d} \sqrt{BT} + n^{3/2} d^5 \sqrt{BT}\right) \\ &= \widetilde{\mathcal{O}}\left(n^{3/2} d^3 \binom{d+n}{d} \sqrt{BT}\right), \end{aligned}$$

concluding the proof. □

We observe that the regret bound in Theorem 4.1 has an exponential dependence on the number of states of nature d and the number of receiver's actions n, due to the binomial coefficient. Indeed, when d, respectively n, is constant, the regret bound is of the order of $\widetilde{\mathcal{O}}(n^d\sqrt{T})$, respectively $\widetilde{\mathcal{O}}(d^n\sqrt{T})$. Such a dependence is tight, as shown by the lower bounds that we provide in Section 4.4.

4.3.1 Phase 1: `Build-Search-Space`

Given an $\epsilon \in (0, 1)$ and a number of rounds T_1, the `Build-Search-Space` procedure (Algorithm 4.2) computes a subset $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ of normalized slices satisfying two crucial properties needed by the learning algorithm to attain the desired guarantees. The first property is that any slice $x \in \mathcal{X}_\epsilon$ can be "induced" with sufficiently high probability by a signaling scheme, while the second one is that, if $x \notin \mathcal{X}_\epsilon$, then signaling schemes "induce" such a slice with sufficiently small probability. Intuitively, the first property ensures that it is possible to associate any $x \in \mathcal{X}_\epsilon$ with the action $a(x)$ in a number of rounds of the order of $1/\epsilon$, while the second property is needed to bound the loss in sender's expected utility due to only considering signaling schemes with slices in $\mathcal{X}_\epsilon$.

Algorithm 4.2 works by observing realized states of nature θ^t for T_1 rounds, while committing to any signaling scheme meanwhile. This allows the algorithm

Algorithm 4.2 `Build-Search-Space`

Require: $\epsilon \in (0,1)$, number of rounds T_1
1: $\widetilde{\Theta} \leftarrow \varnothing$
2: **while** $t \leq T_1$ **do**
3: Commit to any ϕ_t, observe θ^t, and send s^t
4: Observe feedback a^t and receive u_t^{s}
5: $t \leftarrow t+1$
6: **end while**
7: Compute prior estimate $\widehat{\mu}_t$
8: $\widehat{\mu} \leftarrow \widehat{\mu}_t$
9: **for** $\theta \in \Theta$ **do**
10: **if** $\widehat{\mu}_\theta > 2\epsilon$ **then**
11: $\widetilde{\Theta} \leftarrow \widetilde{\Theta} \cup \{\theta\}$
12: **end if**
13: **end for**
14: $\mathcal{X}_\epsilon \leftarrow \left\{x \in \mathcal{X} \mid \sum_{\theta \in \widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\right\}$
15: **return** $\mathcal{X}_\epsilon$

to build a sufficiently accurate estimate $\widehat{\mu}$ of the true prior μ. Then, the algorithm uses such an estimate to build the set $\mathcal{X}_\epsilon$. Specifically, it constructs $\mathcal{X}_\epsilon$ as the set containing all the normalized slices that are "inducible" with probability at least 2ϵ under the estimated prior $\widehat{\mu}$, after filtering out all the states of nature whose estimated probability is *not* above the 2ϵ threshold (see Lines 9–14 in Algorithm 4.2).

The following lemma establishes the two crucial properties guaranteed by Algorithm 4.2. Intuitively, the lemma follows from applying the multiplicative version of the Chernoff bound, which allows us to distinguish whether prior probabilities μ_θ lie above or below suitable thresholds within $O(1/\epsilon)$ rounds.

Specifically, we show that after T_1 rounds, and with probability at least $1-\delta$, the set $\widetilde{\Theta}$ in the definition of $\mathcal{X}_\epsilon$ excludes all states $\theta \in \Theta$ with $\mu_\theta \leq \epsilon$, while including all states with sufficiently large μ_θ. This directly establishes properties (i) and (ii) of Lemma 4.1. The use of a multiplicative Chernoff bound is essential, since standard concentration inequalities would require $O(1/\epsilon^2)$ rounds, yielding a suboptimal regret bound in terms of T.

Lemma 4.1. *Given* $T_1 := \left\lceil \frac{12}{\epsilon} \log\left(2d/\delta\right) \right\rceil$ *and* $\epsilon \in (0, 1/6d)$, *Algorithm 4.2 employs* T_1 *rounds and terminates with a set* $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ *such that, with probability at least* $1-\delta$: *(i)* $\sum_{\theta \in \Theta} \mu_\theta x_\theta \geq \epsilon$ *for every slice* $x \in \mathcal{X}_\epsilon$ *and (ii)* $\sum_{\theta \in \Theta} \mu_\theta x_\theta \leq 10\epsilon$ *for every slice* $x \in \mathcal{X} \setminus \mathcal{X}_\epsilon$.

Proof. For each $\theta \in \Theta$ we consider two possible cases.

1. If $\mu_\theta > \epsilon$, then we employ the multiplicative Chernoff inequality as follows:

$$\mathbb{P}\left(|\mu_\theta - \widehat{\mu}_\theta| \geq \frac{1}{2}\mu_\theta\right) \leq 2e^{-\frac{T_1 \mu_\theta}{12}},$$

where $T_1 \in \mathbb{N}_+$ is the number of rounds employed to estimate $\widehat{\mu}_\theta$. As

a result, by setting the number of rounds to estimate μ_θ equal to $T_1 = \lceil {}^{12}/_{\epsilon} \log ({}^{2d}/_{\delta}) \rceil$ we get:

$$\mathbb{P}\left(|\mu_\theta - \widehat{\mu}_\theta| \geq \frac{1}{2}\mu_\theta\right) \leq 2\left(\frac{\delta}{2d}\right)^{\frac{\mu_\theta}{\epsilon}} \leq \frac{\delta}{d},$$

since $\mu_\theta > \epsilon$. Then, we get:

$$\mathbb{P}\left(\frac{\mu_\theta}{2} \leq \widehat{\mu}_\theta \leq \frac{3\mu_\theta}{2}\right) \geq 1 - \frac{\delta}{d}. \tag{4.1}$$

Consequently, with a probability of at least $1-\delta/d$ the estimator $\widehat{\mu}_\theta$ is such that $\widehat{\mu}_\theta \in [{}^{\mu_\theta}/_2, {}^{3\mu_\theta}/_2]$. Thus, if $\mu_\theta \geq 6\epsilon$, then $\widehat{\mu}_\theta \geq 3\epsilon > 2\epsilon$ and $\theta \in \widetilde{\Theta}$. We also notice that there always exits a $\theta \in \Theta$ such that $\mu_\theta \geq {}^1/_d \geq 6\epsilon$, since $\epsilon \in (0, {}^1/_{6d})$. Consequently, with a probability of at least $1 - \delta/d$, there always exist a $\theta \in \widetilde{\Theta}$.

2. If $\mu_\theta \leq \epsilon$, then we employ the multiplicative Chernoff inequality as follows:

$$\mathbb{P}\left(\widehat{\mu}_\theta \geq (1+c)\mu_\theta\right) \leq e^{-\frac{c^2 T_1 \mu_\theta}{2+c}},$$

with $c = \epsilon/\mu_\theta$. Thus, by setting the number of rounds employed to estimate μ_θ equal to $T_1 = \lceil {}^{12}/_{\epsilon} \log ({}^{2d}/_{\delta}) \rceil$, we get:

$$\begin{aligned}
\mathbb{P}\left(\widehat{\mu}_\theta \geq \mu_\theta + \epsilon\right) &\leq \exp\left(-\frac{\frac{\epsilon^2}{\mu_\theta^2}\left(\frac{12}{\epsilon}\log\left(\frac{2d}{\delta}\right)\right)\mu_\theta}{2+\frac{\epsilon}{\mu_\theta}}\right) \\
&\leq \exp\left(-\frac{\frac{12\epsilon}{\mu_\theta}\log\left(\frac{2d}{\delta}\right)}{2+\frac{\epsilon}{\mu_\theta}}\right) \\
&\leq \left(\frac{\delta}{2d}\right)^4 \leq \frac{\delta}{d},
\end{aligned}$$

since $x/(x+2) \geq 1/3$, for each $x \geq 1$. As a result, we have:

$$\mathbb{P}\left(\widehat{\mu}_\theta \leq \mu_\theta + \epsilon\right) \geq 1 - \frac{\delta}{d}.$$

Thus, with a probability of at least $1-\delta/d$, if $\mu_\theta \leq \epsilon$, then $\widehat{\mu}_\theta \leq \mu_\theta + \epsilon$, which implies that $\widehat{\mu}_\theta \leq 2\epsilon$. Furthermore, if $\mu_\theta \leq \epsilon$, then $\widehat{\mu}_\theta \leq 2\epsilon$ and $\theta \notin \widetilde{\Theta}$.

Thus, by employing a union bound over the set of natures, we have that if $\mu_\theta \leq \epsilon$, then its corresponding estimate $\widehat{\mu}_\theta$ falls within the interval $[0, 2\epsilon]$ and $\theta \notin \widetilde{\Theta}$, while if $\mu_\theta \geq 6\epsilon$, then its corresponding estimate is such that $\widehat{\mu}_\theta > 2\epsilon$ and

$\theta \in \widetilde{\Theta}$, with a probability of at least $1-\delta$. We also notice that, with the same probability, the set $\widetilde{\Theta}$ is always non empty.

Consequently, for each slice $x \in \mathcal{X}_\epsilon$ with respect to a signal s, the probability of observing s can be lower bounded as follows:

$$\epsilon \leq \frac{1}{2}\sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \leq \frac{3}{4}\sum_{\theta\in\widetilde{\Theta}} \mu_\theta x_\theta \leq \sum_{\theta\in\widetilde{\Theta}} \mu_\theta x_\theta \leq \sum_{\theta\in\Theta} \mu_\theta x_\theta,$$

where the inequalities above hold because of the definition of $\mathcal{X}_\epsilon$ and observing that each $\theta \in \widetilde{\Theta}$ satisfies Equation (4.1) with probability at least $1-\delta$.

Furthermore, for each $x \notin \mathcal{X}_\epsilon$, the two following conditions hold:

$$\frac{1}{2}\sum_{\theta\in\widetilde{\Theta}} \mu_\theta x_\theta \leq \sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \leq 2\epsilon \quad \text{and} \quad \sum_{\theta\notin\widetilde{\Theta}} \mu_\theta x_\theta \leq 6\epsilon \sum_{\theta\notin\widetilde{\Theta}} x_\theta \leq 6\epsilon,$$

with probability at least $1-\delta$. Thus, by putting the two inequalities above together, for each $x \notin \mathcal{X}_\epsilon$, we have:

$$\sum_{\theta\in\Theta} \mu_\theta x_\theta \leq 10\epsilon,$$

with probability at least $1-\delta$, concluding the proof. □

For ease of presentation, we introduce the following *clean event* for Phase 1 of Algorithm 4.1. This encompasses all the situations in which Algorithm 4.2 outputs a set $\mathcal{X}_\epsilon$ with the desired properties.

Definition 4.2 (Phase 1 clean event)**.** *$\mathcal{E}_1$ is the event in which $\mathcal{X}_\epsilon$ meets properties (i)–(ii) in Lemma 4.1.*

4.3.2 Phase 2: `Find-Slices`

Given a set $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ computed by the `Build-Search-Space` procedure and $\zeta \in (0,1)$ as inputs, the `Find-Slices` procedure (Algorithm 4.4) correctly terminates in a suitably-bounded number of rounds with probability at least $1-\zeta$ (see Lemma 4.3), providing as output a $\mathcal{Y} \coloneqq (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$ where $\mathcal{V}$ is the set of all the vertices of polytopes $\mathcal{X}_\epsilon(a_i)$ with $\text{vol}(\mathcal{X}_\epsilon(a_i)) > 0$ and $(a(x))_{x\in\mathcal{V}}$ is a tuple specifying the action $a(x)$ played by the receiver in each slice $x \in \mathcal{V}$. To achieve this goal, the procedure works by searching over the space of normalized slices $\mathcal{X}_\epsilon$, so as to learn *exactly* all the separating hyperplanes H_{ij} characterizing the needed vertices.

The algorithm does so by using and extending tools that have been developed for a related learning problem in Stackelberg games (see Chapter 3). However, one of the main differences with Stackelberg games is that here, given a slice $x \in \mathcal{X}_\epsilon$, the sender does not have a direct way to determine the receiver's best response $a(x)$. Therefore, a crucial component of the `Find-Slices` procedure is a tool to "query" a normalized slice $x \in \mathcal{X}_\epsilon$ in order to obtain the action $a(x)$. This is done by using a subprocedure that we call `Action-Oracle` (see Algorithm 4.3), which works by committing to a signaling scheme including slice x until the signal

corresponding to such a slice is actually sent. Under the clean event $\mathcal{E}_1$, the set $\mathcal{X}_\epsilon$ is built in such a way that `Action-Oracle` returns the desired action $a(x)$ in a number of rounds of the order of $1/\epsilon$, with high probability. This is made formal by the following lemma.

Lemma 4.2. *Under event $\mathcal{E}^1$, given any $\rho \in (0,1)$ and a normalized slice $x \in \mathcal{X}_\epsilon$, if the sender commits to a signaling scheme $\phi : \Theta \to \mathcal{S} := \{s_1, s_2\}$ such that $\phi_\theta(s_1) = x_\theta$ for all $\theta \in \Theta$ during $q := \left\lceil \frac{1}{\epsilon} \log(1/\rho) \right\rceil$ rounds, then, with probability at least $1-\rho$, signal s_1 is sent at least once.*

Proof. In the following, we let τ be the first round in which the sender commits to ϕ. The probability of observing the signal s_1 at a given round $t \geq \tau$ is can be lower bounded as follows:

$$\mathbb{P}\left(s^t = s_1\right) = \sum_{\theta \in \Theta} \mu_\theta x_\theta \geq \epsilon,$$

where the inequality holds under the event $\mathcal{E}_1$. Thus, at each round, the probability of sampling the signal $s_1 \in \mathcal{S}$ is greater or equal to $\epsilon > 0$. Consequently, the probability of never observing the signal $s_1 \in \mathcal{S}$ in q rounds is given by:

$$\mathbb{P}\left(\bigcap_{t=\tau}^{\tau+q-1} \{s^t \neq s_1\}\right) \leq (1-\epsilon)^q \leq \rho,$$

where the last inequality holds by taking

$$q = \left\lceil \frac{\log(\rho)}{\log(1-\epsilon)} \right\rceil \leq \left\lceil \frac{\log(1/\rho)}{\epsilon} \right\rceil,$$

for each $\epsilon \in (0,1)$. As a result, the probability of observing the signal s_1 at least once in q rounds is greater or equal to:

$$\mathbb{P}\left(\bigcup_{t=\tau}^{\tau+q-1} \{s^t \neq s_1\}\right) = 1 - \mathbb{P}\left(\bigcap_{t=\tau}^{\tau+q-1} \{s^t \neq s_1\}\right) \geq 1-\rho,$$

concluding the proof. □

Notice that the signaling scheme used to "query" an $x \in \mathcal{X}_\epsilon$ only employs *two* signals. One is associated with slice x, while the other is crafted so as to make the signaling scheme well defined.

Equipped with Algorithm 4.3, we now describe how the `Find-Slices` procedure (Algorithm 4.4) works.[4] First, at Line 2, Algorithm 4.4 initializes a set $\mathcal{C}$ that will contain all actions that have been "closed" in the sense that their discovered upper bound $\mathcal{U}_\epsilon(a)$ coincides with the entire region $\mathcal{X}_\epsilon(a)$.

Then, at Line 4, Algorithm 4.4 samples a normalized slice x^{int} uniformly at random from the interior of one of the polytopes $\mathcal{X}_\epsilon(a_j)$ that have not yet been closed and queries it, observing the best response $a_j \in \mathcal{A}$. Then, it initializes $\mathcal{X}_\epsilon(a_j)$ as the

[4] While the estimator $\widehat{\mu}_t$ is updated throughout Algorithm 4.4, the space $\mathcal{X}_\epsilon = \{x \in \mathcal{X} \mid \sum_{\theta \in \widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\}$ remains fixed during the execution of such procedure.

Algorithm 4.3 `Action-Oracle`

Require: $x \in \mathcal{X}_\epsilon$
1: $\phi_\theta(s_1) \leftarrow x_\theta$
2: $\phi_\theta(s_2) \leftarrow 1 - x_\theta \; \forall \theta \in \Theta$
3: **do**
4: Commit to $\phi^t = \phi$, observe θ^t, and send s^t
5: Observe feedback a^t
6: $a \leftarrow a^t$
7: **while** $s^t \neq s_1$
8: **return** a

Algorithm 4.4 `Find-Slices`

Require: Parameter $\zeta \in (0, 1)$
1: $\delta \leftarrow \zeta / 2n^2(2(d+n)+n)$
2: $\mathcal{C} \leftarrow \varnothing$
3: **while** $\bigcup_{a_j \in \mathcal{C}} \mathcal{U}(a_j) \neq \mathcal{X}_\epsilon$ **do**
4: $x^{\text{int}} \leftarrow$ Sample a point from $\text{int}\big(\mathcal{X}_\epsilon \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}(a_k)\big)$
5: $a_j \leftarrow$ `Action-Oracle`(x^{int})
6: $\mathcal{U}(a_j) \leftarrow \mathcal{X}_\epsilon$
7: $\mathcal{V} \leftarrow V(\mathcal{U}(a_j))$
8: $B_x \leftarrow$ Bit-complexity of x^{int}
9: $\lambda \leftarrow d 2^{-d(B_x + 4(B + B_\epsilon + B_{\widehat{\mu}})) - 1}$
10: **while** $\mathcal{V} \neq \varnothing$ **do**
11: $v \leftarrow$ Take any vertex in $\mathcal{V}$
12: $x \leftarrow \lambda x^{\text{int}} + (1 - \lambda) v$
13: $a \leftarrow$ `Action-Oracle`(x)
14: **if** $a \neq a_j$ **then**
15: $H_{jk} \leftarrow$ `Find-Hyperplane`$(a_j, \mathcal{U}(a_j), x^{\text{int}}, v, \delta)$
16: $\mathcal{U}(a_j) \leftarrow \mathcal{U}(a_j) \cap \mathcal{H}_{jk}$
17: $\mathcal{V} \leftarrow V(\mathcal{U}(a_j))$
18: **else**
19: $\mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$
20: **end if**
21: **end while**
22: $\mathcal{C} \leftarrow \mathcal{C} \cup \{a_j\}$
23: **end while**
24: $\mathcal{V} \leftarrow \bigcup_{a_j \in \mathcal{C}} V(\mathcal{U}(a_j))$
25: **return** $\mathcal{V}, (\text{`Action-Oracle`}(x))_{x \in \mathcal{V}}$

upper bound $\mathcal{U}(a_j)$ of the region $\mathcal{X}_\epsilon(a_j)$. As a further step, to check whether the upper bound $\mathcal{U}(a_j)$ coincides with $\mathcal{X}_\epsilon(a_j)$, Algorithm 4.4 queries at Line 11 one of the vertices of $\mathcal{U}(a_j)$. Since the same vertex may belong to the intersection of multiple regions, the `Action-Oracle` procedure is called on a suitable convex combination of x^{int} and the vertex v itself (Line 12). If the vertex does not belong to $\mathcal{X}_\epsilon(a_j)$, then a new separating hyperplane is computed (Line 15) and the upper bound $\mathcal{U}(a_j)$ is updated accordingly. This separating hyperplane is obtained through the `Find-Hyperplane` procedure, presented in Appendix B, and is in-

spired by the one developed in Chapter 3 for Stackelberg games.

In this way, the upper bound $\mathcal{U}(a_j)$ is refined by successively adding separating hyperplanes until it coincides with the polytope $\mathcal{X}_\epsilon(a_j)$. Finally, this procedure is repeated for all receiver's actions $a_j \in \mathcal{A}$ such that $\text{vol}(\mathcal{X}_\epsilon(a_j)) > 0$.

The following lemma provides the guarantees of Algorithm 4.4. In Appendix B we report its proof and the details of the different subprocedures executed by Algorithm 4.4.

Lemma 4.3. *Given inputs $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ and $\zeta \in (0,1)$ for Algorithm 4.4, let $L \coloneqq B + B_\epsilon + B_{\widehat{\mu}}$, where B_ϵ, and $B_{\widehat{\mu}}$ denote the bit-complexity of ϵ, and $\widehat{\mu}$, respectively. Then, under event $\mathcal{E}_1$ and with at probability at least $1-\zeta$, Algorithm 4.4 outputs a $\mathcal{Y} \coloneqq (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$ such that*

$$\mathcal{V} = \bigcup_{a_i \in \mathcal{A}:\text{vol}(\mathcal{X}_\epsilon(a_i))>0} V(\mathcal{X}_\epsilon(a_i))$$

in a number of rounds T_2 satisfying:

$$T_2 \leq \widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\zeta}\right)\left(d^7 L + \binom{d+n}{d}\right)\right).$$

We conclude the section by introducing a second clean event, under which Algorithm 4.4 has correctly identified the vertices of the regions $\mathcal{X}_\epsilon(a)$ for all $a \in \mathcal{A}$ such that $\text{vol}(\mathcal{X}_\epsilon(a)) > 0$. As we will further discuss, this event will be crucial to compute approximately optimal signaling schemes.

Definition 4.3 (Phase 2 clean event)**.** *$\mathcal{E}_2$ is the event in which*

$$\mathcal{V} = \bigcup_{a_i \in \mathcal{A}:\text{vol}(\mathcal{X}_\epsilon(a_i))>0} V(\mathcal{X}_\epsilon(a_i)).$$

4.3.3 Phase 3: `Compute-Signaling`

Given $\mathcal{Y} \coloneqq (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$ returned by `Find-Slices` and $\widehat{\mu}_t \in \Delta_\Theta$, Algorithm 4.5 (`Compute-Signaling`) outputs an approximately-optimal signaling scheme by solving an LP (Program (4.2)). This optimizes over weight vectors α with components α_x indexed by the vertices $x \in \mathcal{V}$.

Specifically, it maximizes a weighted sum—with weights α_x—of sender's approximate utilities under slices $x \in \mathcal{V}$. Additionally, a feasible weight vector has to satisfy some consistency constraints ensuring that it is possible to recover a valid signaling scheme from them. Finally, by letting $\alpha^\star$ be an optimal solution to Program (4.2), the signaling scheme ϕ returned by Algorithm 4.5 employs a signal s^x for every $x \in \mathcal{V}$ with $\alpha^\star_x > 0$, plus an additional signal $s^\star$.

The slice of ϕ with respect to a signal s^x is set to be a version of x "scaled" by α_x, while the slice of $s^\star$ is set so that the components of ϕ_θ sum to one. In the following, we prove the guarantees, in terms of sender's expected utility, provided by the signaling scheme ϕ returned by Algorithm 4.5. First, we prove a preliminary lemma showing that any $x \in \mathcal{X}_\epsilon$ can be decomposed as a convex combination of vertexes in $\mathcal{V}$.

Algorithm 4.5 `Compute-Signaling`

Require: $\mathcal{Y} := (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$, $\mathcal{X}_\epsilon$, $\widehat{\mu}_t \in \Delta_\Theta$

1: Solve Program (4.2) for an optimal solution $\alpha^\star$:

$$\max_{\alpha\geq\mathbf{0}} \sum_{x\in\mathcal{V}} \alpha_x \sum_{\theta\in\Theta} \widehat{\mu}_{t,\theta} x_\theta u^{\mathrm{s}}_\theta(a(x)) \quad \text{s.t.} \qquad (4.2)$$
$$\sum_{x\in\mathcal{V}} \alpha_x x_\theta \leq 1 \quad \forall\theta\in\Theta$$

2: $\mathcal{S} \leftarrow \{s^\star\} \cup \{s^x \mid x \in \mathcal{V} \wedge \alpha^\star_x > 0\}$

3: **for** $\theta \in \Theta$ **do**

4: $$\phi_\theta \leftarrow \begin{cases} \phi_\theta(s^x) = \alpha_x x_\theta \quad \forall x \in \mathcal{V} : \alpha^\star_x > 0 \\ \phi_\theta(s^\star) = 1 - \sum_{x\in\mathcal{V}} \alpha_x x_\theta \end{cases}$$

5: **end for**

6: **return** ϕ

Lemma 4.4. *Under event $\mathcal{E}_2$, for every $x \in \mathcal{X}_\epsilon$, there exists an $\alpha \in \Delta_\mathcal{V}$ such that:*

$$x_\theta = \sum_{x'\in\mathcal{V}} \alpha_{x'} x'_\theta \quad \forall\theta\in\Theta.$$

Furthermore, the following holds:

$$\sum_{\theta\in\Theta} \mu_\theta x_\theta u^{\mathrm{s}}_\theta(a(x)) \leq \sum_{x'\in\mathcal{V}} \alpha_{x'} \sum_{\theta\in\Theta} \mu_\theta x'_\theta u^{\mathrm{s}}_\theta(a(x')).$$

Proof. We consider two cases:

1. Let $a(x) = a$. If $x \in \mathcal{X}_\epsilon(a)$ with $\mathrm{vol}(\mathcal{X}_\epsilon(a)) > 0$, then by Carathéodory theorem there exists an $\alpha \in \Delta_{V(\mathcal{X}_\epsilon(a))}$ such that:

$$\sum_{x'\in V(\mathcal{X}_\epsilon(a))} \alpha_{x'} x'_\theta = x_\theta \ \ \forall\theta\in\Theta.$$

Furthermore:

$$\begin{aligned}\sum_{\theta\in\Theta} \mu_\theta x_\theta u^{\mathrm{s}}_\theta(a) &= \sum_{\theta\in\Theta} \mu_\theta \left(\sum_{x'\in V(\mathcal{X}_\epsilon(a))} \alpha_{x'} x'_\theta \right) u^{\mathrm{s}}_\theta(a). \\ &= \sum_{\theta\in\Theta} \sum_{x'\in V(\mathcal{X}_\epsilon(a))} \mu_\theta \alpha_{x'} x'_\theta u^{\mathrm{s}}_\theta(a) \\ &\leq \sum_{\theta\in\Theta} \sum_{x'\in V(\mathcal{X}_\epsilon(a))} \mu_\theta \alpha_{x'} x'_\theta u^{\mathrm{s}}_\theta(a(x')),\end{aligned}$$

we observe that the above decomposition is well defined since, under the event $\mathcal{E}_2$, all the vertices $V(\mathcal{X}_\epsilon(a)) \subseteq \mathcal{V}$ have been identified.

2. Let $a(x) = a_k$. If $x \in \mathcal{X}_\epsilon(a_k)$ with $\text{vol}(\mathcal{X}_\epsilon(a_k)) = 0$, then we show that there exists a $\mathcal{X}_\epsilon(a_j)$ with $\text{vol}(\mathcal{X}_\epsilon(a_j)) > 0$ such that x lies on the boundary of $\mathcal{X}_\epsilon(a_j)$. To observe that, we assume by contradiction that there exists an action $a_j \in \mathcal{A}$ with $\text{vol}(\mathcal{X}_\epsilon(a_j)) > 0$ such that $x \in \text{int}(\mathcal{X}_\epsilon(a_j))$. Then, a_k is a best-response in the set $\mathcal{X}_\epsilon(a_j) \cap \mathcal{H}_{kj}$ with positive volume, reaching a contradiction with $\text{vol}(\mathcal{X}_\epsilon(a_k)) = 0$. Then, $x \in \mathcal{X}_\epsilon(a_j)$ for some $\mathcal{X}_\epsilon(a_j)$ with strictly positive volume, and thus, $x \in H_{jk}$, since $x \in \mathcal{X}_\epsilon(a_j) \cap \mathcal{X}_\epsilon(a_k)$.

   Moreover, we have that $\mathcal{X}_\epsilon(a_j) \cap H_{jk} \subseteq \mathcal{X}_\epsilon(a_k)$. This holds since in any point of the set $\mathcal{X}_\epsilon(a_j) \cap H_{jk}$ it must be the case that a_j is a best response, and the receiver's utility by playing actions a_j is equal to the one of action a_k by definition of H_{jk}.

   It is easy to see that $V(\mathcal{X}_\epsilon(a_j) \cap H_{jk}) \subseteq V(\mathcal{X}_\epsilon(a_j))$. Furthermore, by the convexity of $\mathcal{X}_\epsilon(a_j) \cap H_{jk}$ and the fact that leader's expected utility is linear over $\mathcal{X}_\epsilon(a_j) \cap H_{jk}$ when follower's best response is fixed to a_k, we conclude that there exists an $\alpha \in \Delta_{V(\mathcal{X}_\epsilon(a_j) \cap H_{jk})}$ such that:

$$\sum_{x' \in V(\mathcal{X}_\epsilon(a_j) \cap H_{jk})} \alpha_{x'} x'_\theta = x_\theta \;\; \forall \theta \in \Theta,$$

   and,

$$\begin{aligned}\sum_{\theta \in \Theta} \mu_\theta x_\theta u^{\text{s}}_\theta(a_k) &= \sum_{\theta \in \Theta} \mu_\theta \left(\sum_{x' \in V(\mathcal{X}_\epsilon(a_j) \cap H_{jk})} \alpha_{x'} x'_\theta \right) u^{\text{s}}_\theta(a_k). \\ &= \sum_{\theta \in \Theta} \sum_{x' \in V(\mathcal{X}_\epsilon(a_j) \cap H_{jk})} \mu_\theta \alpha_{x'} x'_\theta u^{\text{s}}_\theta(a_k) \\ &\leq \sum_{\theta \in \Theta} \sum_{x' \in \mathcal{V}} \mu_\theta \alpha_{x'} x'_\theta u^{\text{s}}_\theta(a(x')).\end{aligned}$$

   Where the above decomposition is well defined under the event $\mathcal{E}_2$, since all the vertices $V(\mathcal{X}_\epsilon(a_j) \cap H_{jk}) \subseteq \mathcal{V}$ have been identified.

The two points above conclude the proof. □

We observe that an optimal signaling scheme ϕ can always be represented as a set of normalized slices $x^a \in \mathcal{X}$, each scaled by suitable weights $\alpha^a > 0$, with a ranging over an appropriate subset of receiver actions. By Lemma 4.4, each slice $x^a \in \mathcal{X}_\epsilon$ can be decomposed into a convex combination of the vertices in $\mathcal{V}$ without any loss in the sender's utility. This, in turn, shows that the sender's expected utility for the signaling scheme returned by Algorithm 4.5 is close to OPT. Indeed, an optimal signaling scheme may comprise slices $x^a \notin \mathcal{X}_\epsilon$, but these provide only limited utility to the sender under $\mathcal{E}_1$. This is made formal in the following lemma.

Lemma 4.5. *Given a set of vertices $\mathcal{V}$ returned by Algorithm 4.4, under events $\mathcal{E}_1$ and $\mathcal{E}_2$, the signaling scheme ϕ' returned by Algorithm 4.5 when $\widehat{\mu}_t = \mu$ provides the sender with an expected utility $u^{\text{s}}(\phi') \geq \text{OPT} - 10\epsilon nd$.*

Proof. In the following we let $\phi_\theta \in \Delta_{\mathcal{A}}$ for each $\theta \in \Theta$ be an optimal signaling scheme, where we assume, without loss of generality, such signaling scheme to be direct, meaning that $\mathcal{S} = \mathcal{A}$. Furthermore, for each action $a \in \text{supp}(\phi)$, we define:

$$x_\theta^a = \frac{\phi_\theta(a)}{\sum_{\theta \in \Theta} \phi_\theta(a)} \ \forall \theta \in \Theta \quad \text{and} \quad \alpha_{x^a} = \sum_{\theta \in \Theta} \phi_\theta(a).$$

We observe that each $x^a \in \mathcal{X}(a)$, indeed we have:

$$\begin{aligned}
\sum_{\theta \in \Theta} \mu_\theta x_\theta u_\theta(a) &= \frac{1}{\alpha_{x^a}} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(a) u_\theta(a) \\
&\geq \frac{1}{\alpha_{x^a}} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(a) u_\theta(a') \\
&= \sum_{\theta \in \Theta} \mu_\theta x_\theta u_\theta(a') \ \forall a' \in \mathcal{A}.
\end{aligned}$$

We define the subset $\mathcal{A}' \subseteq \mathcal{A}$ in a way that if $a \in \mathcal{A}'$, then $x^a \in \mathcal{X}_\epsilon$, *i.e.*,

$$\mathcal{A}' \coloneqq \{a \in \mathcal{A} \mid x^a \in \mathcal{X}_\epsilon\}.$$

Then, we let $\mathcal{A}'' \coloneqq \text{supp}(\phi) \setminus \mathcal{A}'$. Furthermore, for each $a \in \mathcal{A}'$, thanks to Lemma 4.4, there exists a distribution $\alpha^a \in \Delta_{\mathcal{V}}$ such that:

$$x_\theta^a = \sum_{x' \in \mathcal{V}} \alpha_{x'}^a x'_\theta,$$

and the following holds:

$$\begin{aligned}
\sum_{\theta \in \Theta} \mu_\theta x_\theta^a u_\theta^{\text{s}}(a) &= \sum_{\theta \in \Theta} \mu_\theta \left(\sum_{x' \in \mathcal{V}} \alpha_{x'}^a x'_\theta \right) u_\theta^{\text{s}}(a). \\
&= \sum_{\theta \in \Theta} \sum_{x' \in \mathcal{V}} \mu_\theta \alpha_{x'}^a x'_\theta u_\theta^{\text{s}}(a) \\
&\leq \sum_{\theta \in \Theta} \sum_{x' \in \mathcal{V}} \mu_\theta \alpha_{x'}^a x'_\theta u_\theta^{\text{s}}(a(x')). \qquad (4.3)
\end{aligned}$$

We also define $\alpha^\star \in \mathbb{R}_+^{|\mathcal{V}|}$ as follows:

$$\alpha_{x'}^\star = \sum_{a \in \mathcal{A}'} \alpha_{x'}^a \alpha_{x^a},$$

for each $x' \in \mathcal{V}$. First, we show that $\alpha^\star \in \mathbb{R}_+^{|\mathcal{V}|}$ is a feasible solution to LP (4.2). Indeed, for each $\theta \in \Theta$, it holds:

$$\sum_{x' \in \mathcal{V}} \alpha_{x'}^\star x'_\theta = \sum_{x' \in \mathcal{V}} \sum_{a \in \mathcal{A}'} \alpha_{x'}^a \alpha_{x^a} x'_\theta$$

$$
\begin{aligned}
&= \sum_{a \in \mathcal{A}'} \alpha_{x^a} \sum_{x' \in \mathcal{V}} \alpha^a_{x'} x'_\theta \\
&= \sum_{a \in \mathcal{A}'} \alpha_{x^a} x^a_\theta \\
&= \sum_{a \in \mathcal{A}'} \phi_\theta(a) \leq 1
\end{aligned}
$$

Where the equalities above holds thanks to Equation (4.3) and the definition of $\alpha^\star$. Then, we show that the utility achieved by $\alpha^\star$ is at least OPT $- 10\epsilon d$. Formally, we have:

$$
\begin{aligned}
\sum_{x' \in \mathcal{V}} \alpha^\star_{x'} \sum_{\theta \in \Theta} \mu_\theta x'_\theta u^{\mathrm{s}}_\theta(a(x')) &= \sum_{x' \in \mathcal{V}} \sum_{a \in \mathcal{A}'} \alpha_{x^a} \alpha^a_{x'} \sum_{\theta \in \Theta} \mu_\theta x'_\theta u^{\mathrm{s}}_\theta(a(x')) \\
&= \sum_{a \in \mathcal{A}'} \alpha_{x^a} \sum_{x' \in \mathcal{V}} \sum_{\theta \in \Theta} \mu_\theta \alpha^a_{x'} x'_\theta u^{\mathrm{s}}_\theta(a(x')) \\
&\geq \sum_{a \in \mathcal{A}'} \alpha_{x^a} \sum_{\theta \in \Theta} x^a_\theta \mu_\theta u^{\mathrm{s}}_\theta(a) \\
&= \sum_{a \in \mathcal{A}'} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(a) u^{\mathrm{s}}_\theta(a) \\
&= \mathrm{OPT} - \sum_{a \in \mathcal{A}''} \sum_{\theta \in \Theta} \mu_\theta \phi_\theta(a) u^{\mathrm{s}}_\theta(a) \\
&= \mathrm{OPT} - \sum_{a \in \mathcal{A}''} \sum_{\theta \in \Theta} \mu_\theta \alpha_{x^a} x^a_\theta u^{\mathrm{s}}_\theta(a) \\
&\geq \mathrm{OPT} - d \sum_{a \in \mathcal{A}''} \sum_{\theta \in \Theta} \mu_\theta x^a_\theta u^{\mathrm{s}}_\theta(a) \\
&\geq \mathrm{OPT} - 10\epsilon n d.
\end{aligned}
$$

Where the first inequality holds thanks to Equation (4.3), the second inequality holds since $\alpha_{x^a} \leq d$ and the last inequality holds since, for each $x \notin \mathcal{X}_\epsilon$, it holds

$$
\sum_{\theta \in \Theta} \mu_\theta x^a_\theta(a) \leq 10\epsilon,
$$

under the event $\mathcal{E}_1$ and $x^a \notin \mathcal{X}_\epsilon$ for every $a \in \mathcal{A}''$. Consequently, $\alpha^\star$ is a feasible solution to LP (4.2) and provides utility of at least OPT $- 10\epsilon n d$.

Given a feasible solution α to LP (4.2), we show how to recover a signaling scheme achieving greater or equal utility. To do so, we define:

$$
\phi' = \begin{cases} \phi'_\theta(s^x) = \alpha_x x_\theta \quad \forall x \in \mathrm{supp}(\alpha), \\ \phi'_\theta(s^\star) = 1 - \sum_{x' \in \mathcal{V}} \alpha_{x'} x'_\theta. \end{cases}
$$

With the same calculation above, it is easy to verify that the signaling scheme ϕ' achieve utility greater or equal to the one achieved by α. □

Finally, thanks to Lemma 4.5, we can prove the following lemma.

Lemma 4.6. *Given inputs $\mathcal{Y} \coloneqq (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$, $\mathcal{X}_\epsilon$, and $\widehat{\mu}_t \in \Delta_\Theta$ for Algorithm 4.5, under events $\mathcal{E}_1$ and $\mathcal{E}_2$, the signaling scheme ϕ output by the algorithm is $\mathcal{O}(\epsilon nd + \nu)$-optimal for $\nu \leq \left|\sum_{\theta\in\Theta} \widehat{\mu}_{t,\theta} - \mu_\theta\right|$.*

Proof. In the following, let ϕ^{LP} denote the optimal solution to LP (4.2) with $\widehat{\mu} = \widehat{\mu}_t$, and let ϕ' denote the optimal solution to LP (4.2) with $\widehat{\mu} = \mu$. Then the following holds:

$$\begin{aligned}
u(\phi^{\text{LP}}) &= \sum_{s\in\mathcal{S}}\sum_{\theta\in\Theta} \mu_\theta \phi^{\text{LP}}_\theta(s) u^{\text{s}}_\theta(a^\phi(s)) \\
&\geq \sum_{s\in\mathcal{S}}\sum_{\theta\in\Theta} \widehat{\mu}_\theta \phi^{\text{LP}}_\theta(s) u^{\text{s}}_\theta(a^\phi(s)) - \left|\sum_{\theta\in\Theta} \widehat{\mu}_\theta - \mu_\theta\right| \\
&\geq \sum_{s\in\mathcal{S}}\sum_{\theta\in\Theta} \widehat{\mu}_\theta \phi'_\theta(s) u^{\text{s}}_\theta(a^\phi(s)) - \left|\sum_{\theta\in\Theta} \widehat{\mu}_\theta - \mu_\theta\right| \\
&\geq \sum_{s\in\mathcal{S}}\sum_{\theta\in\Theta} \mu_\theta \phi'_\theta(s) u^{\text{s}}_\theta(a^\phi(s)) - 2\left|\sum_{\theta\in\Theta} \widehat{\mu}_\theta - \mu_\theta\right| \\
&= u(\phi') - 2\left|\sum_{\theta\in\Theta} \widehat{\mu}_\theta - \mu_\theta\right| \\
&\geq \text{OPT} - 10\epsilon nd - 2\left|\sum_{\theta\in\Theta} \widehat{\mu}_\theta - \mu_\theta\right|.
\end{aligned}$$

Where the last step holds thanks to Lemma 4.5, which holds under the event $\mathcal{E}_1 \cap \mathcal{E}_2$, concluding the proof. □

4.4 Lower bounds for online Bayesian persuasion

In this section, we present two lower bounds on the regret attainable in the setting faced by Algorithm 4.1. The first lower bound shows that an exponential dependence in the number of states of nature d and the number of receiver's actions n is unavoidable. This shows that one cannot get rid of the binomial coefficient in the regret bound of Algorithm 4.1 provided in Theorem 4.1. Formally:

Theorem 4.2. *For any sender's algorithm, there exists a Bayesian persuasion instance in which $n = d + 2$ and the regret R_T suffered by the algorithm is at least $2^{\Omega(d)}$, or, equivalently, $2^{\Omega(n)}$.*

Proof. In the following, for the sake of presentation, we consider a set of instances characterized by an even number $d \in \mathbb{N}_+$ of states of nature and $n = d+2$ receiver actions. All instances share the same uniform prior distribution and the same sender utility, given by $u^{\text{s}}_\theta(a_{d+1}) = 1$ for all $\theta \in \Theta$, and $u^{\text{s}}_\theta(a) = 0$ for all $\theta \in \Theta$ and for all $a \in \mathcal{A} \setminus \{a_{d+1}\}$. Each instance is parameterized by a vector p

belonging to a set $\mathcal{P}$ defined as follows:

$$\mathcal{P} \coloneqq \left\{ p \in \{0,1\}^d \;\middle|\; \sum_{i=1}^{d} p_i = \frac{d}{2} \right\}.$$

Furthermore, we assume that the receiver utilities in each instance I_p are:

$$(I_p) \begin{cases} u_{\theta_i}(a_j) = \delta_{ij}, & \forall i, j \in [d], \\ u_{\theta_i}(a_{d+1}) = \frac{2}{d} p_i, & \forall i \in [d], \\ u_{\theta_i}(a_{d+2}) = \frac{2}{d}, & \forall i \in [d]. \end{cases}$$

We show that $\xi'_{\theta_i} \coloneqq \frac{2}{d} p_i$ for each $i \in [d]$ is the only posterior inducing the receiver action $a_{d+1} \in \mathcal{A}$ in the instance I_p, since the receiver breaks ties in favor of the sender. To prove this, we observe that action a_{d+1} is preferred to action a_{d+2} only in those posteriors that satisfy the condition $\xi_{\theta_i} = 0$ for each $i \in [d]$ such that $p_i = 0$. Furthermore, to incentivize action a_{d+1} over the set of actions a_i with $i \in [d]$, the following condition must hold:

$$\sum_{i \in [d] : p_i > 0} \xi_{\theta_i} u_{\theta_i}(a_{d+1}) = \frac{2}{d} \sum_{i \in [d] : p_i > 0} \xi_{\theta_i} = \frac{2}{d} \geq \max_{i \in [d] : p_i > 0} \xi_{\theta_i}.$$

We note that the last step holds only if $\xi_{\theta_i} \leq 2/d$ for each i such that $p_i > 0$. Consequently, since the number of $\xi_{\theta_i} > 0$ is at most $d/2$, it follows that $\xi_{\theta_i} = 2/d$ for each i with $p_i > 0$. Thus, the only posterior-inducing action a_{d+1} is given by $\xi'_{\theta_i} \coloneqq \frac{2}{d} p_i$. Moreover, given $p \in \mathcal{P}$, an optimal signaling scheme γ is defined as $\gamma(\xi') = 1/2$ and $\gamma(\xi'') = 1/2$, with $\xi'' = \mu - \frac{1}{2}\xi'$. With a simple calculation, it is possible to verify that the expected sender utility under γ is equal to $1/2$.

We set the time horizon $T = \lfloor |\mathcal{P}|/4 \rfloor$ to show that any algorithm suffers regret of at least $2^{\Omega(d)}$. This is sufficient to prove the statement. We start by making the following simplifications about the behavior of the algorithm. First, we observe that if the algorithm can choose any posterior (instead of a signaling scheme), then this will only increase its performance. Consequently, we assume that the algorithm chooses a posterior ξ_t at each round $t \in [T]$.

Thus, we can apply Yao's minimax principle to show that any deterministic algorithm fails against a distribution over instances. In the following, we consider a uniform distribution over instances I_p with $p \in \mathcal{P}$. Furthermore, we observe that the feedback of any algorithm is actually binary. Thus, it is easy to see that an optimal algorithm works as follows: (i) it ignores the feedback whenever the action is not a_{d+1}, and (ii) it starts to play the optimal posterior when the action is a_{d+1}, since it has found an optimal posterior.

This observation is useful for showing that any deterministic algorithm does not find a posterior that induces action a_{d+1} with probability at least $1 - |\mathcal{P}|/(4|\mathcal{P}|) = 3/4$ (since it can choose only $\lfloor |\mathcal{P}|/4 \rfloor$ posteriors among the $|\mathcal{P}|$ possible optimal posteriors). Hence, by Yao's minimax principle, for any (randomized) algorithm there exists an instance such that the regret suffered in the T rounds is at

least:

$$R_T \geq \frac{3}{4}\frac{T}{2} \geq \frac{1}{4}\left\lfloor \frac{|\mathcal{P}|}{4} \right\rfloor \geq \frac{|\mathcal{P}|}{32},$$

since $\lfloor x \rfloor \geq x - 1 \geq x/2$ for each $x \geq 2$. Finally, we note that $|\mathcal{P}| = \binom{d}{d/2} = 2^{\Omega(d)}$, which concludes the proof. □

The second lower bound shows that the regret bound attained by Algorithm 4.1 is tight in T.

Theorem 4.3. *For any sender's algorithm, there exists a Bayesian persuasion instance in which the regret R_T suffered by the algorithm is at least $\Omega(\sqrt{T})$.*

Proof. To prove the theorem, we introduce two instances characterized by two states of nature and four receiver's actions. In both the instances the sender's utility is given by $u^{\mathrm{s}}_{\theta}(a_1) = u^{\mathrm{s}}_{\theta}(a_2) = 0$ for all $\theta \in \Theta$, while $u^{\mathrm{s}}_{\theta_1}(a_3) = 1$, $u^{\mathrm{s}}_{\theta_2}(a_3) = 0$ and $u^{\mathrm{s}}_{\theta_2}(a_4) = 0$ $u^{\mathrm{s}}_{\theta_2}(a_4) = 1$. The receiver's utilities and the prior distributions in the two instances are:

① $\begin{cases} \mu^1_{\theta_1} = \frac{1}{2} + \epsilon,\ \mu^1_{\theta_2} = \frac{1}{2} - \epsilon \\ u^1_{\theta_1}(a_1) = \frac{1}{(2+4\epsilon)},\ u^1_{\theta_2}(a_1) = \frac{1}{(10-20\epsilon)} \\ u^1_{\theta_1}(a_2) = \frac{1}{(10+20\epsilon)},\ u^1_{\theta_2}(a_2) = \frac{1}{(2-4\epsilon)} \\ u^1_{\theta_1}(a_3) = \frac{3}{10},\ u^1_{\theta_2}(a_3) = \frac{3}{10} \\ u^1_{\theta_1}(a_4) = 0,\ u^1_{\theta_2}(a_4) = \frac{1}{(2-4\epsilon)} \end{cases}$ ② $\begin{cases} \mu^2_{\theta_1} = \frac{1}{2} - \epsilon,\ \mu^2_{\theta_2} = \frac{1}{2} + \epsilon \\ u^2_{\theta_1}(a_1) = \frac{1}{(2-4\epsilon)},\ u^2_{\theta_2}(a_1) = \frac{1}{(10+20\epsilon)} \\ u^2_{\theta_1}(a_2) = \frac{1}{(10-20\epsilon)},\ u^2_{\theta_2}(a_2) = \frac{1}{(2+4\epsilon)} \\ u^2_{\theta_1}(a_3) = \frac{3}{10},\ u^2_{\theta_2}(a_3) = \frac{3}{10} \\ u^2_{\theta_1}(a_4) = 0,\ u^2_{\theta_2}(a_4) = \frac{1}{(2+4\epsilon)} \end{cases}$

with $\epsilon \in (0, 1/4)$. With a simple calculation, we can show that, in both the two instances, for any signaling scheme ϕ employing a generic set of signals $\mathcal{S}$, the sender receives the following feedback:

1. For each $s \in \mathcal{S}$ such that $\phi_{\theta_1}(s) > \phi_{\theta_2}(s)$, then $a^{\phi}(s) = a_1$.
2. For each $s \in \mathcal{S}$ such that $0 < \phi_{\theta_1}(s) < \phi_{\theta_2}(s)$, then $a^{\phi}(s) = a_2$.
3. For each $s \in \mathcal{S}$ such that $\phi_{\theta_1}(s) = \phi_{\theta_2}(s)$, then $a^{\phi}(s) = a_3$.
4. For each $s \in \mathcal{S}$ such that $0 = \phi_{\theta_1}(s)$, then $a^{\phi}(s) = a_4$.

As a result, for any signaling scheme the sender may commit to, the feedback in each signal is the same. Thus, we assume without loss of generality that the sender commits only to signaling schemes that maximize the probability of inducing actions a_3 or a_4. This is because the sender gains no information from committing to one signaling scheme over another, while only the signaling schemes that induce these two actions yield strictly positive expected utility.

Furthermore, we can restrict our attention to direct signaling schemes, *i.e.*, those in which the set of signals coincides with the set of actions. Thus, at each

round, we assume that the sender commits to a signaling scheme ϕ^t of the form:

$$\phi^t := \begin{cases} \phi^t_{\theta_1}(a_3) = \phi^t_{\theta_2}(a_3) := \phi^t_1, \\ \phi^t_{\theta_1}(a_4) = 0,\ \phi^t_{\theta_2}(a_4) = 1 - \phi^t_1, \\ \phi^t_{\theta_1}(a_1) = 1 - \phi^t_1,\ \phi^t_{\theta_2}(a_1) = 0, \end{cases} \tag{4.4}$$

with $\phi^t_1 \in [0, 1]$. We also notice that, at each round, the optimal signaling scheme in the first instance is the one that induces action a_3, meaning $\phi^t_1 = 1$ for each $t \in [T]$. While the optimal signaling scheme in the second instance at each round is the one that reveals the state of nature to the receiver, meaning $\phi^t_1 = 0$ for each $t \in [T]$. In such a way, the learning task undertaken by the sender reduces to select a value of $\phi^t_1 \in [0, 1]$ for each $t \in [T]$ controlling the probability of inducing action a_3 over action a_4.

In the following, we define $\mathbb{P}^1$ and $\mathbb{P}^2$ as the probability measures induced by running a fixed algorithm on the first and second instance, respectively, and we let $\mathbb{E}^1$ and $\mathbb{E}^2$ denote expectation under these measures. The cumulative regret in the first instance can be written as follows:

$$\begin{aligned} R^1_T &= \mathbb{E}^1 \left[\sum_{t=1}^{T} \left(\frac{1}{2} + \epsilon - \phi^t_1 \left(\frac{1}{2} + \epsilon \right) - \left(1 - \phi^t_1\right) \left(\frac{1}{2} - \epsilon \right) \right) \right] \\ &= 2\epsilon \mathbb{E}^1 \left[\sum_{t=1}^{T} \left(1 - \phi^t_1\right) \right]. \end{aligned}$$

Similarly, in the second instance, the cumulative regret is given by:

$$R^2_T = 2\epsilon \mathbb{E}^2 \left[\sum_{t=1}^{T} \phi^t_1 \right].$$

Furthermore, it is easy to check that:

$$R^1_T \geq \mathbb{P}^1 \left(\sum_{t=1}^{T} \phi^1_t \leq T/2 \right) \epsilon T \quad \text{and} \quad R^2_T \geq \mathbb{P}^2 \left(\sum_{t=1}^{T} \phi^1_t \geq T/2 \right) \epsilon T.$$

By employing the relative entropy identities divergence decomposition we also have that:

$$\mathcal{KL}\left(\mathbb{P}^1, \mathbb{P}^2\right) = T \cdot \mathcal{KL}\left(\mu^1, \mu^2\right) \leq \frac{64}{3} T \epsilon^2 \leq 22 T \epsilon^2,$$

where we employed the fact that for two Bernoulli distributions it holds

$$\mathcal{KL}(\mathrm{Be}(p), \mathrm{Be}(q)) \leq \frac{(p-q)^2}{q(1-q)}.$$

Then, by employing the Bretagnolle-Huber inequality we have that,

$$\begin{aligned} R_T^1 + R_T^2 &\geq \epsilon T \left(\mathbb{P}^1 \left(\sum_{t=1}^{T} \phi_t^1 \leq T/2 \right) + \mathbb{P}^2 \left(\sum_{t=1}^{T} \phi_t^1 \geq T/2 \right) \right) \\ &\geq \frac{1}{2} \epsilon T \exp\left(-\mathcal{KL}\left(\mathbb{P}^1, \mathbb{P}^2 \right) \right) \\ &\geq \frac{1}{2} \epsilon T \exp\left(-22 T \epsilon^2 \right). \end{aligned}$$

By taking $\epsilon = \sqrt{\frac{1}{22T}}$, we get $R_T^1 + R_T^2 \geq C_1 \sqrt{T}$, with $C_1 = {e^{-1}}/{2\sqrt{22}}$.
Thus:

$$R_T^1 \geq \frac{C_1}{2}\sqrt{T} \quad \vee \quad R_T^2 \geq \frac{C_1}{2}\sqrt{T},$$

concluding the proof. □

4.5 The sample complexity of Bayesian persuasion

In this section, we show how the no-regret learning algorithm developed in Section 4.3 can be easily adapted to solve a related *Bayesian persuasion PAC-learning problem*. Specifically, given an (additive) approximation error $\beta \in (0,1)$ and a probability $\eta \in (0,1)$, the goal of such a problem is to determine a β-optimal signaling scheme with probability at least $1-\eta$, by using the minimum possible number of rounds. This can be also referred to as the *sample complexity* of learning signaling schemes. As in the regret-minimization problem addressed in Section 4.3, we assume that the sender does *not* know anything about both the prior distribution μ and receiver's utility function u. We address the Bayesian

Algorithm 4.6 `PAC-Persuasion-w/o-Clue`

Require: $\beta \in (0,1), \eta \in (0,1)$
1: $\delta \leftarrow \eta/2, \zeta \leftarrow \eta/2, \epsilon_1 \leftarrow \beta/12nd, t \leftarrow 1, \epsilon \leftarrow 1$
2: **while** $\epsilon \geq \epsilon_1$ **do**
3: $\quad \epsilon \leftarrow \epsilon/2$
4: **end while**
5: $T_1 \leftarrow \left\lceil \frac{1}{2\epsilon^2} \log\left(\frac{2d}{\delta}\right) \right\rceil$
6: $\mathcal{X}_\epsilon \leftarrow$ `Build-Search-Space`(T_1, ϵ)
7: $\mathcal{Y} \leftarrow$ `Find-Slices`$(\mathcal{X}_\epsilon, \zeta)$
8: $\phi \leftarrow$ `Compute-Signaling`$(\mathcal{Y}, \mathcal{X}_\epsilon, \widehat{\mu})$
9: **return** ϕ

persuasion PAC-learning problem by modifying Algorithm 4.1, as given in Algorithm 4.6. The first two phases of the algorithm follow the line of Algorithm 4.1, with the `Build-Search-Space` and `Find-Slices` procedures being called for suitably-defined parameters ϵ, δ, T_1, and ζ (taking different values with respect to their counterparts in Algorithm 4.1). In particular, the value of ϵ depends on β and is carefully chosen to control the bit complexity of the quantities employed in the `Find-Slices` procedure. Specifically, at Line 2, the parameter ϵ, initially

set to one, is iteratively halved until $\epsilon \in [\epsilon_1, 2\epsilon_1]$. This step keeps the bit complexity of the parameter ϵ sufficiently small as required by the `Find-Slices` procedure to be efficiently employed. Finally, in its third phase, the algorithm calls `Compute-Signaling` to compute a signaling scheme ϕ that can be proved to β-optimal with probability at least $1 - \eta$.

The most relevant difference between Algorithm 4.6 and Algorithm 4.1 is the number of rounds used to build the prior estimate defining $\mathcal{X}_\epsilon$. Specifically, while the latter has to employ T_1 of the order of $1/\epsilon$ and rely on a multiplicative Chernoff bound to get tight regret guarantees, the former has to use T_1 of the order of $1/\epsilon^2$ and standard concentration inequalities to get an $\mathcal{O}(\epsilon)$-optimal solution.

Formally:

Lemma 4.7. *Given $T_1 := \lceil 1/2\epsilon^2 \log(2d/\delta) \rceil$ and $\epsilon \in (0,1)$, Algorithm 4.2 employs T_1 rounds and outputs $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ such that, with probability at least $1-\delta$, the following holds:*

(i) $\sum_{\theta \in \Theta} \mu_\theta x_\theta \geq \epsilon$ *for every slice* $x \in \mathcal{X}_\epsilon$,

(ii) $\sum_{\theta \in \Theta} \mu_\theta x_\theta \leq 6\epsilon$ *for every slice* $x \in \mathcal{X} \setminus \mathcal{X}_\epsilon$, *and*

(iii) $|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon$ *for every* $\theta \in \Theta$.

Proof. Thanks to the definition of $T_1 := \lceil 1/2\epsilon^2 \log(2d/\delta) \rceil$ in Algorithm 4.6, and by applying both a union bound and Hoeffding's inequality, we obtain:

$$\mathbb{P}\left(|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon\right) \geq 1 - \delta, \quad \forall \theta \in \Theta.$$

Consequently, if $|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon$ for each $\theta \in \Theta$, then for each $x \in \mathcal{X}_\epsilon$, the probability of inducing the slice x can be lower bounded as follows:

$$\epsilon \leq \sum_{\theta \in \Theta'} \widehat{\mu}_\theta x_\theta - \epsilon \leq \sum_{\theta \in \Theta'} (\widehat{\mu}_\theta - \epsilon) x_\theta \leq \sum_{\theta \in \Theta'} \mu_\theta x_\theta \leq \sum_{\theta \in \Theta} \mu_\theta x_\theta$$

where the above inequalities hold because each $x \in \mathcal{X}_\epsilon$ satisfies the constraint

$$\sum_{\theta \in \Theta'} \widehat{\mu}_\theta x_\theta \geq 2\epsilon.$$

Furthermore, if $|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon$ for each $\theta \in \Theta$, then for each $x \notin \mathcal{X}_\epsilon$ we have:

$$\sum_{\theta \in \Theta'} \mu_\theta x_\theta \leq \epsilon + \sum_{\theta \in \Theta'} (\mu_\theta - \epsilon) x_\theta \leq \epsilon + \sum_{\theta \in \Theta'} \widehat{\mu}_\theta x_\theta \leq 3\epsilon, \tag{4.5}$$

since, if $x \notin \mathcal{X}_\epsilon$, it holds $\sum_{\theta \in \Theta'} \widehat{\mu}_\theta x_\theta \leq 2\epsilon$, and,

$$\sum_{\theta \notin \Theta'} \mu_\theta x_\theta \leq \sum_{\theta \notin \Theta'} (\mu_\theta - \epsilon) x_\theta + \epsilon \leq \sum_{\theta \notin \Theta'} \widehat{\mu}_\theta x_\theta + \epsilon \leq 3\epsilon. \tag{4.6}$$

Thus, by combining Equation (4.5) and Equation (4.6), we have:

$$\sum_{\theta \in \Theta} \mu_\theta x_\theta \leq 6\epsilon,$$

when $x \notin \mathcal{X}_\epsilon$, concluding the proof. □

By Lemma 4.7, it is possible to show that the event $\mathcal{E}^1$ holds. Hence, the probability that a signaling scheme including a slice $x \in \mathcal{X}_\epsilon$ actually "induces" such a slice is at least ϵ, and, thus, the results concerning the second phase of Algorithm 4.1 are valid also in this setting.

Finally, whenever the events $\mathcal{E}_1$ and $\mathcal{E}_2$ hold, we can provide an upper bound on the number of rounds required by Algorithm 4.6 to compute a β-optimal signaling scheme as desired. Formally:

Theorem 4.4. *Given $\beta \in (0,1)$ and $\eta \in (0,1)$, with probability at least $1-\eta$, Algorithm 4.6 outputs a β-optimal signaling scheme in a number of rounds T such that:*

$$T \leq \widetilde{\mathcal{O}}\left(\frac{n^2}{\beta^2}\log^2\left(\frac{1}{\eta}\right)\left(d^8 B + d\binom{d+n}{d}\right)\right).$$

Proof. Thanks to Lemma 4.7, with probability at least $1-\eta/2$, Algorithm 4.6 correctly completes Phase 1 in $T_1 = \mathcal{O}(1/\epsilon^2 \log(1/\eta)\log(d))$ rounds. Thus, with probability at least $1-\eta/2$, both the event $\mathcal{E}_1$ and the inequalities $|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon$ for all $\theta \in \Theta$ hold. Consequently, under $\mathcal{E}_1$, with probability at least $1-\eta/2$, Algorithm 4.6 partitions the search space $\mathcal{X}_\epsilon$ in at most

$$\widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\eta}\right)\left(d^7 L + \binom{d+n}{d}\right)\right)$$

rounds, as stated in Lemma 4.3. Moreover, $L = B + B_\epsilon + B_{\widehat{\mu}}$, with

$$B_{\widehat{\mu}} = \mathcal{O}(\log(T_1)) = \mathcal{O}\left(\log(1/\epsilon) + \log(1/\eta) + \log(d)\right).$$

As a result, with probability at least $1-\eta$, Algorithm 4.6 terminates after a number of rounds N that can be upper bounded as follows

$$N \leq \widetilde{\mathcal{O}}\left(\frac{1}{\epsilon^2}\log\left(\frac{1}{\eta}\right)\log(d) + \frac{n^2}{\epsilon}\log^2\left(\frac{1}{\eta}\right)\left(d^7(B+B_\epsilon) + \binom{d+n}{d}\right)\right).$$

Furthermore, if $|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon$ for each $\theta \in \Theta$, then

$$\left|\sum_{\theta \in \Theta}(\widehat{\mu}_\theta - \mu_\theta)\right| \leq \sum_{\theta \in \Theta}|\widehat{\mu}_\theta - \mu_\theta| \leq \epsilon d.$$

Consequently, by Lemma 4.6, with probability at least $1-\eta$, Algorithm 4.6 computes a $12n\epsilon d$-optimal solution.

Therefore, setting $\epsilon_1 := \beta/12nd$ and choosing $\epsilon \leq \epsilon_1$ in accordance with the way the algorithm operates, we obtain that, with probability at least $1-\eta$, Algorithm 4.6 returns a β-optimal solution in a number of rounds N bounded by

$$\begin{aligned} N &\leq \widetilde{\mathcal{O}}\left(\frac{n^2d^2}{\beta^2}\log\left(\frac{1}{\eta}\right) + \frac{n^3}{\beta}\log^2\left(\frac{1}{\eta}\right)\left(d^8(B+B_\epsilon) + d\binom{d+n}{d}\right)\right) \\ &= \widetilde{\mathcal{O}}\left(\frac{n^3}{\beta^2}\log^2\left(\frac{1}{\eta}\right)\left(d^8(B+B_\epsilon) + d\binom{d+n}{d}\right)\right) \\ &= \widetilde{\mathcal{O}}\left(\frac{n^3}{\beta^2}\log^2\left(\frac{1}{\eta}\right)\left(d^8B + d\binom{d+n}{d}\right)\right), \end{aligned}$$

where the last equality follows from the fact that $B_\epsilon = \mathcal{O}\left(\log\left(nd/\beta\right)\right)$, which completes the proof. □

Finally, we give two negative results establishing tightness.

Theorem 4.5. *There exist two absolute constants $\kappa, \lambda > 0$ such that no algorithm is guaranteed to return an κ-optimal signaling scheme with probability of at least $1-\lambda$ by employing less than $2^{\Omega(n)}$ and $2^{\Omega(d)}$ rounds, even when the prior distribution μ is known to the sender.*

Proof. In the proof of Theorem 4.2 we show that, with probability $3/4$, in $N = \lfloor |\mathcal{P}|/4 \rfloor$ rounds any algorithm does not correctly identify the posterior inducing action a_{d+1}. This is because, any deterministic algorithm can identify the optimal posterior only in $\lfloor |\mathcal{P}|/4 \rfloor$ instances, as observed in Theorem 4.2.

As a result, in the remaining $|\mathcal{P}| - N$ instances, any deterministic algorithm will receive the same feedback and thus will always output the same posterior after N rounds, which will result in the optimal one in only a single instance.

Thus, thanks to the Yao's minimax principle, there is no algorithm that is guaranteed to return an optimal solution with probability at least:

$$\frac{3}{4}\left(\frac{|\mathcal{P}| - N - 1}{|\mathcal{P}| - N}\right) \geq \frac{3}{8}\left(\frac{|\mathcal{P}| - N}{|\mathcal{P}| - N}\right) = \frac{3}{8}.$$

Finally, we observe that $\text{OPT} = 1/2$, while any algorithm that does not induce the posterior ξ' provides an expected utility equal to zero. As a result, for each $\kappa < 1/2$ there is no algorithm that is guaranteed to return a solution which is κ-optimal in $\lfloor |\mathcal{P}|/4 \rfloor = 2^{\Omega(d)}$ rounds with probability at least $3/8$. □

Theorem 4.6. *Given $\beta \in (0, 1/8)$ and $\eta \in (0,1)$, no algorithm is guaranteed to return a β-optimal signaling scheme with probability at least $1-\eta$ by employing less than $\Omega\left(\frac{1}{\beta^2}\log(1/\eta)\right)$ rounds.*

Proof. To prove the theorem, we consider the same instance and the same definitions introduced in the proof of Theorem 4.6. In this case, we let $\mathbb{P}^1$ ($\mathbb{P}^2$) be the probability distribution generated by the execution of a given algorithm in the first (second) instance for $N = \lceil \log(1/4\eta)/22\epsilon^2 \rceil$ rounds, with $\epsilon \in (0, 1/4)$.

Furthermore, we introduce the event $\mathcal{E}$, under which the signaling scheme returned at the round N, according to the definition presented in Equation (4.4), is such that $\phi^N \leq 1/2$. We notice that, if such signaling scheme is such that $\phi^N < 1/2$, then the sender's expected utility in the first instance is smaller or equal to $1/2$, thus being $\epsilon/2$-optimal. At the same time, if $\phi^N \geq 1/2$ in the second instance, then the solution returned by the algorithm is not $\epsilon/2$-optimal.

Thus, by employing the Bretagnolle-Huber inequality we have:

$$\mathbb{P}^1(\mathcal{E}) + \mathbb{P}^2\left(\mathcal{E}^C\right) \geq \frac{1}{2}\exp\left(-\mathcal{KL}\left(\mathbb{P}^1, \mathbb{P}^2\right)\right) \geq \frac{1}{2}\exp\left(-22N\epsilon^2\right),$$

since $\mathcal{KL}\left(\mathbb{P}^1, \mathbb{P}^2\right) \leq 22N\epsilon^2$, as observed in the proof of Theorem 4.6. Finally, by employing the definition of N, we have:

$$\mathbb{P}^1(\mathcal{E}) \geq \eta \quad \vee \quad \mathbb{P}^2\left(\mathcal{E}^C\right) \geq \eta.$$

As a result, by setting $2\beta = \epsilon$, the statement of the lemma holds. □

4.6 Sample complexity with known prior

In this section, we discuss the *Bayesian persuasion PAC-learning problem* when the prior distribution μ is known to the sender. To tackle the problem, we propose Algorithm 4.7. The main difference with respect to Algorithm 4.6 is that, in this case, we do not need to employ the `Build-Search-Space` procedure, as the prior is already known to the sender. This allows us to compute a β-optimal signaling scheme in only $\mathcal{O}(1/\beta)$ rounds, instead of $\mathcal{O}(1/\beta^2)$ rounds as in the case with an unknown prior. In this case, the following theorem holds.

Algorithm 4.7 `Main-Algorithm-SC-Known`

Require: $\eta \in (0,1), \beta \in (0,1), \mu \in \Delta_\Theta$
1: $\epsilon_1 \leftarrow \beta/10nd$, $\widehat{\mu} \leftarrow \mu$
2: **while** $\epsilon \geq \epsilon_1$ **do**
3: $\quad \epsilon \leftarrow \epsilon/2$
4: **end while**
5: $\widetilde{\Theta} \leftarrow \{\theta \in \Theta \mid \widehat{\mu}_\theta > 2\epsilon\}$
6: $\mathcal{X}_\epsilon \leftarrow \left\{x \in \mathcal{X} \mid \sum_{\theta \in \widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\right\}$
7: $\mathcal{Y} \leftarrow$ `Find-Slices`$(\mathcal{X}_\epsilon, \eta)$
8: $\phi \leftarrow$ `Compute-Signaling`$(\mathcal{Y}, \mathcal{X}_\epsilon, \mu)$
9: **return** ϕ

Theorem 4.7. *With probability at least $1-\eta$ and in Algorithm 4.2 computes a β-optimal signaling scheme in a number of rounds equal to:*

$$\widetilde{\mathcal{O}}\left(\frac{n^2}{\beta}\log^2\left(\frac{1}{\eta}\right)\left(d^8 B + d\binom{d+n}{d}\right)\right).$$

Proof. Since $\widehat{\mu} = \mu$, the clean event $\mathcal{E}_1$ holds with probability one. Consequently, thanks to Lemma 4.3, with probability at least $1 - \eta$, Algorithm 4.7 correctly partition the search space $\mathcal{X}_\epsilon$ in at most:

$$\widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\eta}\right)\left(d^7 L + \binom{d+n}{d}\right)\right)$$

rounds, where we let $L = B + B_\epsilon + B_{\widehat{\mu}}$. Furthermore, according to Algorithm 4.7, we have $\epsilon \leq \epsilon_1 := \beta/10nd$. As a result, $L = \mathcal{O}\left(B + \log(nd) + \log(1/\beta)\right)$, since $B_{\widehat{\mu}} \leq B$ and $B_\epsilon = \mathcal{O}\left(\log\left(1/\epsilon_1\right)\right) = \mathcal{O}\left(\log(nd) + \log\left(1/\beta\right)\right)$.

Furthermore, under the event $\mathcal{E}_2$, Algorithm 4.7 computes a $10\epsilon nd$-optimal solution, as guaranteed by Lemma 4.6, with $\widehat{\mu} = \mu$. Thus, with probability at least $1 - \eta$, Algorithm 4.7 returns a β-optimal solution in

$$\widetilde{\mathcal{O}}\left(\frac{n^3}{\beta}\log^2\left(\frac{1}{\eta}\right)\left(d^8 B + d\binom{d+n}{d}\right)\right)$$

rounds, concluding the proof. □

Theorem 4.7 shows that, differently from the case with an unknown prior, it is possible to achieve a $\mathcal{O}\left(\log(\frac{1}{\eta})/\beta\right)$ upper bound with respect to the input parameters $\beta, \eta > 0$. Finally, we show that such a dependence is tight.

Theorem 4.8. *Given $\beta \in (0, 1/5)$ and $\eta \in (0, 1)$ no algorithm is guaranteed to return an β-optimal signaling scheme with probability of at least $1 - \eta$ employing $\Omega\left(\log(\frac{1}{\eta})/\beta\right)$ rounds, even when the prior distribution is known to the sender.*

Proof. We consider two instances characterized by two states of nature and three receiver actions. The two instances share the same prior distribution, defined as $\mu_{\theta_1} = 4\beta$ and $\mu_{\theta_2} = 1 - 4\beta$. In both instances, the sender's utility is given by $u_\theta^{\mathrm{s}}(a_1) = 0$, $u_\theta^{\mathrm{s}}(a_2) = 1/2$, and $u_\theta^{\mathrm{s}}(a_3) = 1$ for all $\theta \in \Theta$. Furthermore, we assume that the receiver's utilities in the two instances are given by:

$$① \begin{cases} u_{\theta_1}(a_1) = 1,\ u_{\theta_2}(a_1) = 1/2 \\ u_{\theta_1}(a_2) = 1/2,\ u_{\theta_2}(a_2) = 1 \\ u_{\theta_1}(a_3) = 1,\ u_{\theta_2}(a_3) = 0 \end{cases} \qquad ② \begin{cases} u_{\theta_1}(a_1) = 1,\ u_{\theta_2}(a_1) = 1/2 \\ u_{\theta_1}(a_2) = 1/2,\ u_{\theta_2}(a_2) = 1 \\ u_{\theta_1}(a_3) = 1/2,\ u_{\theta_2}(a_3) = 0 \end{cases}$$

We observe that the only case in which the sender receives different feedback in the two instances is when they induce the posterior distribution $\xi_1 := (1, 0)$. This is because, when the sender induces ξ_1 in the first instance, the receiver plays action $a_3 \in \mathcal{A}$, breaking ties in favor of the sender, while in the second instance the receiver plays action $a_1 \in \mathcal{A}$. Such a posterior can be induced, in both instances, with probability at most β to remain consistent with the prior.

We also observe that, in the first instance, the optimal sender's signaling scheme γ is such that $\gamma(\xi_1) = 4\beta$ and $\gamma(\xi_2) = 1 - 4\beta$, where we denote $\xi_2 := (0, 1)$. Furthermore, the sender's expected utility under γ is equal to $(1 + 4\beta)/2$. In the second instance, the optimal sender's signaling scheme γ is such that

$\gamma(\xi_2) = 1 - 8\beta$ and $\gamma(\xi_3) = 8\beta$, with $\xi_3 := (1/2, 1/2)$. It is easy to verify that this signaling scheme achieves an expected utility of $1/2$.

We let $\mathbb{P}^1$ and $\mathbb{P}^2$ denote the probability measures induced by the execution of a given algorithm in the first and second instances, respectively. Furthermore, we introduce the event E_N, under which, during the first N rounds, the sender never observes action $a_3 \in \mathcal{A}$. It is easy to verify that this event occurs with probability at least $\mathbb{P}^1(E_N) \geq (1 - 4\beta)^N$ in the first instance. This is because, at each round, action a_3 can be observed with probability at most 4β. In contrast, since in the second instance the receiver never plays a_3, we have $\mathbb{P}^2(E_N) = 1 \geq (1 - 4\beta)^N$.

Then, by letting γ_1^N (γ_2^N) be the signaling scheme returned after N rounds in the first (second) instance, we have:

$$\begin{aligned}
&\mathbb{P}^1\left(\gamma_1^N(\xi_1) \leq 2\beta\right) + \mathbb{P}^2\left(\gamma_2^N(\xi_1) \geq 2\beta\right) \\
&\quad \geq \mathbb{P}^1\left(\gamma_1^N(\xi_1) \leq 2\beta, E_N\right) + \mathbb{P}^2\left(\gamma_2^N(\xi_1) \geq 2\beta, E_N\right) \\
&\quad \geq \mathbb{P}^1\left(\gamma_1^N(\xi_1) \leq 2\beta \,|E_N\right)\mathbb{P}^1\left(E_N\right) + \mathbb{P}^2\left(\gamma_2^N(\xi_1) \geq 2\beta \,|E_N\right)\mathbb{P}^2\left(E_N\right) \\
&\quad \geq \left(\mathbb{P}^1\left(\gamma_1^N(\xi_1) \leq 2\beta \,|E_N\right) + \mathbb{P}^2\left(\gamma_2^N(\xi_1) \geq 2\beta \,|E_N\right)\right)\mathbb{P}^1\left(E_N\right) \\
&\quad = \mathbb{P}^1\left(E_N\right) \\
&\quad \geq (1 - 4\beta)^N \geq 2\eta. \qquad (4.7)
\end{aligned}$$

The above result holds by observing that, under the event E_N, the behavior of any algorithm in the first instance coincides with that of the same algorithm in the second one, as they receive the same feedback. Furthermore, Equation (4.7) holds when N is such that:

$$N \leq \frac{\log(1/2\eta)}{10\beta} \leq \frac{\log(2\eta)}{\log(1 - 4\beta)},$$

if $\beta \leq 1/5$. Finally, we observe that if $\gamma_1^N(\xi_1) \leq 2\beta$, then the sender's expected utility in the first instance is at most $1/2 + \beta$. This is because, for any signaling scheme γ_1^N, we have:

$$u(\gamma_1^N) \leq \gamma_1^N(\xi_1)u^s(\xi_1) + \frac{1}{2}\left(1 - \gamma_1^N(\xi_1)\right) = \frac{1}{2} + \beta.$$

Thus, if $\gamma_1^N(\xi_1) \leq 2\beta$, then the signaling scheme γ_1^N is at most β-optimal.

Equivalently, if the sender's final signaling scheme γ_2^N in the second instance is such that $\gamma_2^N(\xi_1) \geq 2\beta$, then the sender's utility in the second instance is at most $1/2 - \beta$. Thus, if $\gamma_2^N(\xi_1) \geq 2\beta$, the signaling scheme γ_2^N is at most β-optimal. Then, according to Equation (4.7), we either have:

$$\mathbb{P}^1\left(\gamma_1^N(\xi_1) \leq 2\beta\right) \geq \eta \quad \text{or} \quad \mathbb{P}^2\left(\gamma_2^N(\xi_1) \geq 2\beta\right) \geq \eta,$$

showing that if the number of rounds $N \leq \frac{\log(1/2\eta)}{10\beta}$, then there exists an instance such that no algorithm is guaranteed to return a β-optimal signaling scheme with probability at least $1 - \eta$. □

4.7 Related works

There is a rich literature combining Bayesian persuasion with ideas from *online learning*, aimed at relaxing the sender's initial knowledge of the prior or the receiver's utility. However, most existing approaches achieve this only partially. Some works assume access to the prior (see, *e.g.*, [Castiglioni et al., 2020b, 2021b, Bernasconi et al., 2023a]), while others assume knowledge of the receiver's utilities (see, *e.g.*, [Zu et al., 2021, Wu et al., 2022, Lin and Li, 2025]).

The line of work initiated by Castiglioni et al. [2020b] was the first to introduce *online learning* into Bayesian persuasion with the aim of reducing the sender's dependence on knowledge of the receiver's utilities (see also [Castiglioni et al., 2021b, Bernasconi et al., 2023a]). In their framework, uncertainty is modeled through an *adversary* who chooses a receiver's *type* at each round, with the type specifying the utility structure. However, the sender must still know in advance the set of possible types, their associated utilities, and the prior distribution.

In contrast, the line of research by Zu et al. [2021] and Wu et al. [2022] examines repeated Bayesian persuasion settings where the sender learns to disclose *persuasive* recommendations, *i.e.*, suggestions that the receiver is incentivized to follow. This is motivated by the fact that the receiver in their framework may not have access to the prior distribution or to their utility function to compute a best-response action. Lin and Li [2025] instead assume that the receiver has full knowledge of both the prior and the utility function, so that they can compute a best response after observing a signaling scheme. All these algorithms attain sublinear regret while guaranteeing persuasiveness at every round with high probability, despite having no knowledge of the prior distribution.

In addition to the works presented above, the problem of learning optimal signaling schemes in Bayesian persuasion settings has received growing attention over the last few years. Camara et al. [2020] study an adversarial setting where the receiver does not know the prior, and the receiver's behavior is aimed at minimizing internal regret. Feng et al. [2022] study the online Bayesian persuasion problem faced by a platform that observes some relevant information about the state of a product and repeatedly interacts with a population of myopic receivers through a recommendation mechanism. Agrawal et al. [2023] design a regret-minimization algorithm in an advertising setting based on the Bayesian persuasion framework, assuming that the receiver's utility function satisfies some regularity conditions.

CHAPTER 5

Learning to persuade multiple receivers

In this chapter we study Markov Persuasion Processes (MPPs), the natural extension of one-shot Bayesian persuasion framework to multi-state, Markovian settings. MPPs serve as a framework to model applications such as e-commerce and recommendation systems, where a platform (the sender) holds an informational advantage (e.g., view statistics) and leverages it to persuade users (the receivers) to watch suggested content or purchase products, thereby maximizing long-term engagement. Differently from prior work, we focus on settings in which the sender has no knowledge of the transition functions, the prior distributions over outcomes, or the receivers' utility functions.

Our goal is to design learning algorithms that remain persuasive while achieving regret sublinear in the number of episodes T. However, since in our setting both the sender and the receivers have no prior knowledge about transitions, outcomes, or rewards, achieving persuasiveness at every episode—as in [Wu et al., 2022]—is not possible. As a consequence, the natural requirement is that both the regret and the cumulative violation of persuasiveness grow sublinearly in T.

As a warm-up, we first analyze the full-feedback case, where after each episode the sender observes the rewards associated with every possible action in all visited state-outcome pairs. We introduce the Optimistic Persuasive Policy Search (`OPPS`) algorithm and prove that, under full feedback, it achieves $\widetilde{O}(\sqrt{T})$ regret and violation. We then extend `OPPS` to partial feedback by incorporating an exploration phase to gather information about persuasiveness, and prove that the algorithm attains $\widetilde{O}(T^{\alpha})$ regret and $\widetilde{O}(T^{\max\{1-\alpha/2,\alpha\}})$ violation for any $\alpha \in [1/2, 1]$. Finally, we show that this trade-off is tight by establishing matching lower bounds.

5.1 Markov persuasion processes

Markov Persuasion Processes (MPPs) [Wu et al., 2022] generalize the setting studied in Chapter 4 to settings where the sender faces a stream of receivers in an MDP, with each receiver *myopically* taking an action maximizing immediate reward. An (*episodic*) MPP is defined as a tuple $M := \big(X, A, \Theta, \mu, P, \{r_{S,t}\}_{t=1}^{T}, \{r_{R,t}\}_{t=1}^{T}\big)$, where:

- T is the number of episodes.[1]
- X, A, and Θ are finite sets of states, actions, and outcomes, respectively.[2]
- $\mu : X \to \Delta_\Theta$ is a prior function defining a probability distribution over outcomes at each state. We let $\mu(\theta|x)$ be the probability of sampling outcome $\theta \in \Theta$ in state $x \in X$.
- $P : X \times \Theta \times A \to \Delta_X$ is a transition function. We let $P(x'|x, \theta, a)$ be the probability of going from $x \in X$ to $x' \in X$ by taking action $a \in A$, when the outcome in state x is $\theta \in \Theta$.
- $\{r_{S,t}\}_{t=1}^{T}$ is a sequence specifying a sender's reward function $r_{S,t} : X \times \Theta \times A \to [0, 1]$ at each episode t. Given $x \in X$, $\theta \in \Theta$, and $a \in A$, each $r_{S,t}(x, \theta, a)$ for $t \in [T]$ is sampled independently from a distribution $\nu_S(x, \theta, a)$ supported on $[0, 1]$ with mean $r_S(x, \theta, a)$.
- $\{r_{R,t}\}_{t=1}^{T}$ is a sequence defining a receivers' reward function $r_{R,t} : X \times \Theta \times A \to [0, 1]$ at each episode t. Given $x \in X$, $\theta \in \Theta$, and $a \in A$, each $r_{R,t}(x, \theta, a)$ for $t \in [T]$ is sampled independently from a distribution $\nu_R(x, \theta, a)$ supported on $[0, 1]$ with mean $r_R(x, \theta, a)$.

We focus w.l.o.g. on *loop-free* episodic MPPs, as customary in online learning in MDPs (see, *e.g.*, [Rosenberg and Mansour, 2019]). In a loop-free MPP, states are partitioned into $L + 1$ layers $X_0, \dots, X_L$ such that $X_0 := \{x_0\}$ and $X_L := \{x_L\}$, with x_0 being the initial state starting the episode and x_L being the final one, in which the episode ends. Moreover, by letting $\mathcal{K} := [0 \dots L-1]$ for ease of notation, $P(x'|x, \theta, a) > 0$ only when $x' \in X_{k+1}$ and $x \in X_k$ for some $k \in \mathcal{K}$.[3]

At each episode of an episodic MPP, the sender commits to a *signaling policy* $\phi : X \times \Theta \to \Delta_\mathcal{S}$, which defines a probability distribution over a finite set $\mathcal{S}$ of signals for the receivers for every state $x \in X$ and outcome $\theta \in \Theta$. For ease of notation, we denote by $\phi(\cdot|x, \theta) \in \Delta_\mathcal{S}$ such probability distributions, with $\phi(s|x, \theta)$ being the probability of sending a signal $s \in \mathcal{S}$ in state x when the realized outcome is θ. Similarly to one-shot Bayesian persuasion, a myopic receiver acting at state $x \in X$ and receiving signal $s \in \mathcal{S}$ infers a posterior distribution over outcomes and plays a best-response action. We denote by $b^\phi(s, x) \in A$ the best response played by such a receiver under the signaling policy ϕ (assuming ties are broken in favor of the sender).

As customary in Bayesian persuasion (see, *e.g.*, [Arieli and Babichenko, 2019]), a revelation-principle-style argument allows to focus w.l.o.g. on signaling policies

[1] We denote with $[a \dots b]$ the set of all integers from a to b.

[2] In this chapter, we denote the set of states of nature as the set of outcomes Θ, to avoid confusion with the set of states X of the MPP.

[3] The loop-free property is w.l.o.g. since any episodic MPP with finite horizon H that is *not* loop-free can be cast into a loop-free one by duplicating states H times, *i.e.*, $x \in X$ is mapped to new states (x, k) with $k \in [H]$.

that are direct and persuasive. Formally, a signaling policy is *direct* if the set of signals coincides with the set of actions, namely $\mathcal{S} = A$. Intuitively, signals should be interpreted as action recommendations for the receivers. Moreover, a direct signaling policy is said to be *persuasive* if it incentivizes the receivers to follow recommendations. Formally, $\phi : X \times \Theta \to \Delta_A$ is persuasive if for every state $x \in X$ and recommendation $a \in A$,

$$\sum_{\theta \in \Theta} \mu(\theta|x)\phi(a|x,\theta)\big(r_R(x,\theta,a) - r_R(x,\theta,b^\phi(a,x))\big) \geq 0.$$

Intuitively, the inequality above states that a receiver acting at state x is better off following sender's recommendation to play action a, since by doing so they get an (unnormalized) expected reward greater than or equal to what they would obtain by playing a best-response action $b^\phi(a,x)$.

Algorithm 5.1 shows the interaction between sender and receivers at $t \in [T]$. Sender and receivers do *not* know anything about the transition function P, the prior function μ, and the rewards $r_{S,t}(x,\theta,a), r_{R,t}(x,\theta,a)$ (including their distributions). At the end of each episode, the sender gets to know the triplets (x_k,θ_k,a_k)—for all $k \in \mathcal{K}$—that are *visited* during the episode, and an additional *feedback* about rewards. In this work, we consider two types of feedback. The first one—called *full* feedback—encompasses all agents' rewards for the pairs (x_k,θ_k) visited during the episode, *i.e.*, the rewards for all the triplets (x_k,θ_k,a) for $a \in A$.

Algorithm 5.1 Sender-Receivers Interaction at $t \in [T]$

The rewards $r_{S,t}(x,\theta,a), r_{R,t}(x,\theta,a)$ are sampled
Sender publicly commits to $\phi_t : X \times \Theta \to \Delta_A$
The state of the MPP is initialized to x_0
for $k = 0, \dots, L-1$ **do**
    Sender observes outcome $\theta_k \sim \mu(x_k)$
    Sender draws recommendation $a_k \sim \phi(\cdot|x_k,\theta_k)$
    A *new* Receiver observes a_k and plays it
    The MPP evolves to $x_{k+1} \sim P(\cdot|x_k,\theta_k,a_k)$
    Sender observes the next state x_{k+1}
end for
Sender observes *feedback* for every $k \in [0 \dots L-1]$:
- *full* $\to (r_{S,t}(x_k,\theta_k,a), r_{R,t}(x_k,\theta_k,a))_{a \in A}$
- *partial* $\to r_{S,t}(x_k,\theta_k,a_k), r_{R,t}(x_k,\theta_k,a_k)$

The second type—called *partial* feedback—only consists in agents' rewards for the visited triplets (x_k,θ_k,a_k).[4]

Notice that Algorithm 5.1 assumes that receivers play recommended actions. This is standard in settings where the sender has *not* enough information to be persuasive at every episode, and it motivates why learning algorithms are designed to guarantee that the cumulative violation of persuasiveness grows sublinearly in

[4] In this work we use the adjective *full* to refer to a type of feedback that is *not* the most informative one. Indeed, a full feedback according to the classical terminology used in online learning [Cesa-Bianchi and Lugosi, 2006, Orabona, 2019] would encompass agents' rewards for all the possible triplets (x,θ,a), while full feedback in our terminology only consists in the rewards for the triplets with $x = x_k$ and $\theta = \theta_k$ for some $k \in \mathcal{K}$.

T, or, equivalently, that the per-round violation of persuasiveness goes to zero as T grows [Bernasconi et al., 2022, Cacciamani et al., 2023, Gan et al., 2023b]. Indeed, this ensures that it is in the receivers' best interest to stick to recommendations.

5.2 The learning problem

In this section, we formally introduce the learning problem tackled in the rest of this chapter. First, in Section 5.2.1, we extend the notion of occupancy measure to MPPs. In Section 5.2.2, we formally introduce learning objectives. Finally, in Section 5.2.3, we provide some preliminary elements needed by our algorithms, developed in Sections 5.3 and 5.4.

5.2.1 Occupancy measures in MPPs

Next, we extend the well-known notion of *occupancy measure* of an MDP [Rosenberg and Mansour, 2019] to MPPs. Given a transition function P, a signaling policy ϕ, and a prior function μ, the occupancy measure induced by P, ϕ, and μ is a vector $q^{P,\phi,\mu} \in [0,1]^{|X\times\Theta\times A\times X|}$ whose entries are specified as follows. For every $x \in X_k$, $\theta \in \Theta$, $a \in A$, and $x' \in X_{k+1}$ with $k \in \mathcal{K}$, it holds:

$$q^{P,\phi,\mu}(x,\theta,a,x') := \mathbb{P}\Big((x_k,\theta_k,a_k,x_{k+1}) = (x,\theta,a,x') \mid P,\phi,\mu\Big),$$

which is the probability that the next state is x' after the receiver plays action a in state x when the realized outcome is θ, under transition function P, signaling policy ϕ, and prior μ. Moreover, we also let

$$\begin{cases} q^{P,\phi,\mu}(x,\theta,a) := \sum_{x'\in X_{k+1}} q^{P,\phi,\mu}(x,\theta,a,x') \\ q^{P,\phi,\mu}(x,\theta) := \sum_{a\in A} q^{P,\phi,\mu}(x,\theta,a) \\ q^{P,\phi,\mu}(x) := \sum_{\theta\in\Theta} q^{P,\phi,\mu}(x,\theta), \end{cases}$$

for the ease of notation.

The following lemma characterizes the set of *valid* occupancy measures and it is a generalization to the MPP setting of a lemma by Rosenberg and Mansour [2019].

Lemma 5.1. *A vector $q \in [0,1]^{|X\times\Theta\times A\times X|}$ is a valid occupancy measure of an MPP if and only if*

$$(i) \sum_{x\in X_k}\sum_{\theta\in\Theta}\sum_{a\in A}\sum_{x'\in X_{k+1}} q(x,\theta,a,x') = 1 \quad \forall k \in \mathcal{K},$$

$$(ii) \sum_{x'\in X_{k-1}}\sum_{\theta\in\Theta}\sum_{a\in A} q(x',\theta,a,x) = q(x) \quad \forall k \in [1\dots L-1], \forall x \in X_k,$$

$$(iii)\ P^q = P \text{ and } \mu^q = \mu,$$

where P is the transition function of the MPP and μ its prior function, while P^q and μ^q are the transition and prior functions, respectively, induced by q (see definitions below).

As it is the case in standard MDPs, a valid occupancy measure $q \in [0,1]^{|X\times\Theta\times A\times X|}$ induces a transition function P^q and a signaling policy ϕ^q. Moreover, in an MPP, a valid occupancy measure also induces a prior function μ^q. These are defined as follows:

$$P^q(x'|x,\theta,a) := \frac{q(x,\theta,a,x')}{q(x,\theta,a)},\ \phi^q(a|x,\theta) := \frac{q(x,\theta,a)}{q(x,\theta)},\ \mu^q(\theta|x) := \frac{q(x,\theta)}{q(x)}.$$

Thus, using valid occupancy measures is *equivalent* to using signaling policies. In the following, we denote by $\mathcal{Q} \subseteq [0,1]^{|X\times\Theta\times A\times X|}$ the set of all the valid occupancy measures of an MPP.

5.2.2 Learning objectives

Our goal is to design learning algorithms for the sender in an episodic MPP. We would like algorithms that prescribe sequences of signaling policies ϕ_t that maximize sender's cumulative reward over the T episodes, while at the same time guaranteeing that the violation of persuasiveness constraints is bounded. Notice that, differently from Wu et al. [2022], we do *not* aim at designing learning algorithms whose policies ϕ_t are persuasive at every episode t with high probability, since this is unattainable in our setting in which the sender does *not* know anything about the environment (see Theorem 5.6). Thus, in this chapter we pursue a different objective, formally described in the following.

Baseline. First, we introduce the baseline used to evaluate sender's performances. This is defined as the value of the optimization problem faced by the sender in the *offline* version of the MPP. Such a problem is concerned with expectations of the stochastic quantities in the episodic MPP. By exploiting occupancy measures, the problem can be formulated as the following linear program:

$$\max_{q\in\mathcal{Q}} \sum_{x\in X}\sum_{\theta\in\Theta}\sum_{a\in A} q(x,\theta,a) r_S(x,\theta,a) \quad \text{s.t.} \tag{5.1a}$$

$$\sum_{\theta\in\Theta} q(x,\theta,a)\Big(r_R(x,\theta,a) - r_R(x,\theta,a')\Big) \geq 0$$

$$\forall x \in X, \forall \theta \in \Theta, \forall a \in A, \forall a' \neq a \in A. \tag{5.1b}$$

Intuitively, Problem (5.1) computes an occupancy measure (or, equivalently, signaling policy) maximizing sender's expected reward subject to persuasiveness constraints. We let our baseline be defined as

$$\text{OPT} := \sum_{x\in X,\theta\in\Theta,a\in A} r_S(x,\theta,a) q^*(x,\theta,a),$$

where $q^* \in \mathcal{Q}$ denotes an optimal solution to Problem (5.1). In the following, we denote by ϕ^* an optimal signaling policy, defined as $\phi^* := \phi^{q^*}$.

Metrics. We evaluate the performances of learning algorithms by means of two distinct metrics. The first one is the (*cumulative*) *regret* R_T, which accounts for the difference between the cumulative sender's expected reward obtained by always

playing ϕ^* and that achieved by using the signaling policies ϕ_t prescribed by the algorithm. Formally,

$$R_T := T \cdot \text{OPT} - \sum_{t\in[T]} r_S(x,\theta,a) q_t(x,\theta,a)$$
$$= \sum_{t\in[T]} r_S(x,\theta,a)\big(q^*(x,\theta,a) - q_t(x,\theta,a)\big),$$

where we let $q_t := q^{P,\phi_t,\mu}$ be the occupancy measure induced by ϕ_t. The second metric used to evaluate learning algorithms is the (*cumulative*) *violation* V_T, which is formally defined as follows:

$$V_T := \sum_{t\in[T]} \sum_{x\in X} \sum_{\theta\in\Theta} \sum_{a\in A} q_t(x,\theta,a) \cdot \big(r_R(x,\theta,b^{\phi_t}(a,x)) - r_R(x,\theta,a)\big).$$

Intuitively, V_T encodes the overall expected loss in persuasiveness over the T episodes. Our goal is to develop learning algorithms that prescribe signaling policies ϕ_t guaranteeing that both R_T and V_T grow sublinearly in T, namely $R_T = o(T)$ and $V_T = o(T)$.[5]

5.2.3 Estimators and confidence bounds

Before delving in algorithm design, we introduce estimators and confidence bounds for the stochastic quantities involved in an MPP, namely, transitions, priors, sender's rewards, and receivers' ones. In the following, we let $N_t(x,\theta,a,x') \in \mathbb{N}$ be the number of episodes up to $t \in [T]$ (this excluded) in which the tuple (x,θ,a,x') is visited. Formally, $N_t(x,\theta,a,x') := \sum_{\tau\in[t-1]} \mathbb{1}_\tau\{x,\theta,a,x'\}$, where the indicator function is 1 if and only if the tuple is visited at τ. Similarly, we define the counters $N_t(x,\theta,a,)$, $N_t(x,\theta)$, and $N_t(x)$ in terms of their respective indicators $\mathbb{1}_\tau\{x,\theta,a\}$, $\mathbb{1}_\tau\{x,\theta\}$, and $\mathbb{1}_\tau\{x\}$, which are equal to 1 if and only if (x,θ,a), (x,θ), and x, respectively, are visited at τ.

Next, we define the estimators employed by our algorithms. At the beginning of each episode $t \in [T]$, the estimated probability of going from $x \in X$ to $x' \in X$ by playing $a \in A$, when the outcome realized in state x is $\theta \in \Theta$, is equal to

$$\overline{P}_t\,(x'|x,\theta,a) := \frac{N_t(x,\theta,a,x')}{\max\{1, N_t(x,\theta,a)\}}.$$

Moreover, for every $x \in X$ and $\theta \in \Theta$, the estimated probability of sampling outcome θ from the prior probability distribution at state x is defined as

$$\overline{\mu}_t(\theta|x) := \frac{N_t(x,\theta)}{\max\{1, N_t(x)\}}.$$

Finally, for every state $x \in X$, outcome $\theta \in \Theta$, and action $a \in A$, the estimated sender's and receivers' rewards are defined as

$$\overline{r}_{S,t}(x,\theta,a) := \frac{\sum_{\tau\in[t-1]} r_{S,\tau}(x,\theta,a)\mathbb{1}_\tau\{x,\theta,a\}}{\max\{1, N_t(x,\theta,a)\}},$$

[5] We compactly denote the receivers' best response in a given state-action pair $(x,a) \in X \times A$ at time $t \in [T]$ as $b^t(a,x) := b^{\phi_t}(a,x)$.

and

$$\overline{r}_{R,t}(x,\theta,a) := \frac{\sum_{\tau\in[t-1]} r_{R,\tau}(x,\theta,a)\mathbb{1}_\tau\{x,\theta,a\}}{\max\{1, N_t(x,\theta,a)\}}.$$

Appendix C formally defines the confidence bounds employed by our algorithms. For the transition function P, at each $t \in [T]$, for every $x \in X$, $\theta \in \Theta$, and $a \in A$, we provide a confidence bound $\epsilon_t(x,\theta,a)$ for the probability distribution over next states associated with the triplet (x,θ,a), where the distance between distributions is expressed in $\|\cdot\|_1$-norm (see Lemma C.1). Similarly, we provide a confidence bound $\zeta_t(x)$ in terms of $\|\cdot\|_1$-norm for the prior $\mu(x)$ at each state $x \in X$ (see Lemma C.2). Moreover, for every $x \in X$, $\theta \in \Theta$, and $a \in A$, we provide confidence bounds $\xi_{S,t}(x,\theta,a)$ and $\xi_{R,t}(x,\theta,a)$ for sender's and receivers' rewards, respectively, associated with the triplet (x,θ,a) (see Lemmas C.3 and C.4 for the full feedback case, while Lemmas C.5 and C.6 for the partial feedback one). In conclusion, for ease of presentation, for a confidence parameter $\delta \in (0,1)$, we refer to the event in which all the confidence bounds hold—called *clean event*—as $\mathcal{E}(\delta)$. By combining all the lemmas in Appendix C, $\mathcal{E}(\delta)$ holds with probability at least $1-4\delta$ (by applying a union bound).

5.3 The full feedback case

We first address settings with full feedback, as a warm-up towards the analysis of partial feedback.

5.3.1 The `OPPS` algorithm with full feedback

We propose an algorithm called Optimisitc Persuasive Policy Search (`OPPS`). At each episode, the algorithm solves a variation of the offline optimization problem (Problem (5.1)), called `Opt-Opt`, obtained by substituting mean values with upper/lower confidence bounds. Specifically, `Opt-Opt` is *optimistic* with respect to both the sender's rewards and the satisfaction of persuasiveness constraints, and it is presented in Problem (5.9) in Section 5.3.3. Crucially, by using occupancy measures, `Opt-Opt` can be formulated as an LP, and, thus, solved efficiently. Notice that, since confidence bounds for P and μ are expressed in terms of $||\cdot||_1$-norm, in order to formulate `Opt-Opt` as an LP we need some additional variables and linear constraints, as described in detail in Section 5.3.3.

Algorithm 5.2 Optimistic Persuasive Policy Search (*full*)

Require: X, A, T, confidence parameter $\delta \in (0,1)$
1: Initialize all estimators to 0 and all bounds to $+\infty$
2: **for** $t = 1, \dots, T$ **do**
3: Update all estimators $\overline{P}_t, \overline{\mu}_t, \overline{r}_{S,t}, \overline{r}_{R,t}$ and bounds $\epsilon_t, \zeta_t, \xi_{S,t}, \xi_{R,t}$
4: $\widehat{q}_t \leftarrow$ Solve `Opt-Opt` (Problem (5.9))
5: $\phi_t \leftarrow \phi^{\widehat{q}_t}$
6: Run Algorithm 5.1 by committing to ϕ_t
7: Observe *full* feedback from Algorithm 5.1
8: **end for**

Algorithm 5.2 provides the pseudocode of OPPS with *full* feedback. At each $t \in [T]$, the algorithm first updates all the estimators and confidence bounds according to the feedback received in previous episodes (Line 3). Then, it commits to the signaling policy ϕ_t induced by an optimal solution $\widehat{q}_t$ to Opt-Opt, computed in Line 4. Notice that, the occupancy measure q_t resulting from committing to ϕ_t (and used in the definitions of R_T and V_T) is in general different from $\widehat{q}_t$, as the former is defined in terms of the true (and unknown) transition and prior functions, namely P and μ.

5.3.2 Algorithm analysis with full feedback

Next, we prove the guarantees of OPPS with *full* feedback. A crucial component of our analysis is a relation between the occupancy measures $\widehat{q}_t$ computed by the OPPS algorithm and the occupancy measures q_t that actually result from committing to ϕ_t under the true transitions and priors. This is formally stated by the following lemma.[6]

Lemma 5.2. *Given any $\delta \in (0,1)$, under the clean event $\mathcal{E}(\delta)$, with probability at least $1-2\delta$, it holds that*

$$\sum_{t\in[T]} \|q_t - \widehat{q}_t\|_1 \leq \mathcal{O}\left(L^2|X|\sqrt{T|A||\Theta|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}\right).$$

Proof. We start noticing that, for any $(x,\theta,a) \in X \times \Theta \times A$, we have:

$$\begin{aligned}
&\sum_{x'\in X_{k(x)+1}} |q^{P_t,\phi_t,\mu_t}(x,\theta,a,x') - q^{P,\phi_t,\mu}(x,\theta,a,x')| \\
&= \sum_{x'\in X_{k(x)+1}} \left|q^{P_t,\phi_t,\mu_t}(x,\theta,a)P_t(x'|x,\theta,a) - q^{P,\phi_t,\mu}(x,\theta,a)P(x'|x,\theta,a)\right| \\
&\leq \sum_{x'\in X_{k(x)+1}} \left|q^{P_t,\phi_t,\mu_t}(x,\theta,a)P_t(x'|x,\theta,a) - q^{P,\phi_t,\mu}(x,\theta,a)P_t(x'|x,\theta,a)\right| \\
&\quad + \sum_{x'\in X_{k(x)+1}} \left|q^{P,\phi_t,\mu}(x,\theta,a)P_t(x'|x,\theta,a) - q^{P,\phi_t,\mu}(x,\theta,a)P(x'|x,\theta,a)\right| \\
&= \sum_{x'\in X_{k(x)+1}} \left|(q^{P_t,\phi_t,\mu_t} - q^{P,\phi_t,\mu})(x,\theta,a)\right|P_t(x'|x,\theta,a) + q^{P,\phi_t,\mu}(x,\theta,a)\,|(P_t - P)(x'|x,\theta,a)| \\
&\quad = \left|q^{P_t,\phi_t,\mu_t}(x,\theta,a) - q^{P,\phi_t,\mu}(x,\theta,a)\right| + q^{P,\phi_t,\mu}(x,\theta,a)\|P_t(\cdot|x,\theta,a) - P(\cdot|x,\theta,a)\|_1.
\end{aligned}$$

Thus, summing over $t \in [T]$ and $(x,\theta,a) \in X \times \Theta \times A$ we obtain:

$$\begin{aligned}
&\sum_{t\in[T]} \|q_t - \widehat{q}_t\|_1 \\
&\leq \sum_{t\in[T]} \sum_{x,\theta,a} \Big(\left|(q^{P_t,\phi_t,\mu_t} - q^{P,\phi_t,\mu})(x,\theta,a)\right| + q^{P,\phi_t,\mu}(x,\theta,a)\|(P_t - P)(\cdot|x,\theta,a)\|_1 \Big).
\end{aligned}$$

[6] With a slight abuse of notation, we write $(f+g)(x)$ to denote $f(x)+g(x)$.

Next, we focus on the first part of the equation, noticing that:

$$|(q^{P_t,\phi_t,\mu_t} - q^{P,\phi_t,\mu})(x,\theta,a)| \leq \left|(q^{P_t,\phi_t,\mu_t} - q^{P_t,\phi_t,\mu})(x,\theta,a)\right| + \left|(q^{P_t,\phi_t,\mu} - q^{P,\phi_t,\mu})(x,\theta,a)\right|$$

Bound on $\left|q^{P_t,\phi_t,\mu_t}(x,\theta,a) - q^{P_t,\phi_t,\mu}(x,\theta,a)\right|$**.** We bound this term by induction. At the first layer we have:

$$\begin{aligned}
&\sum_{x_0 \in X_0} \sum_{\theta_0 \in \Theta} \sum_{a_0 \in A} \left|q^{P_t,\phi_t,\mu_t}(x_0,\theta_0,a_0) - q^{P_t,\phi_t,\mu}(x_0,\theta_0,a_0)\right| \\
&\qquad = \sum_{\theta_0 \in \Theta} \sum_{a_0 \in A} |\mu_t(x_0,\theta_0)\phi_t(a_0|x_0,\theta_0) - \mu(x_0,\theta_0)\phi_t(a_0|x_0,\theta_0)| \\
&\qquad \leq \sum_{\theta_0 \in \Theta} |\mu_t(x_0,\theta_0) - \mu(x_0,\theta_0)| \\
&\qquad = q^{P_t,\phi_t,\mu}(x_0) \sum_{\theta_0 \in \Theta} |\mu_t(x_0,\theta_0) - \mu(x_0,\theta_0)| \,.
\end{aligned}$$

observing that $X_0 = \{x_0\}$. Now we show that, if the result holds for x_{k-1}, it holds for x_k, as follows,

$$\begin{aligned}
&\sum_{x_k \in X_k} \sum_{\theta_k \in \Theta} \sum_{a_k \in A} \left|q^{P_t,\phi_t,\mu_t}(x_k,\theta_k,a_k) - q^{P_t,\phi_t,\mu}(x_k,\theta_k,a_k)\right| \\
&= \sum_{\substack{x_{k-1},\theta_{k-1},a_{k-1} \\ x_k,\theta_k,a_k}} |q^{P_t,\phi_t,\mu_t}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu_t(x_k,\theta_k) + \\
&\qquad - q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu(x_k,\theta_k)|\phi_t(a_k|x_k,\theta_k) \\
&= \sum_{\substack{x_{k-1},\theta_{k-1},a_{k-1} \\ x_k,\theta_k}} |q^{P_t,\phi_t,\mu_t}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu_t(x_k,\theta_k) + \\
&\qquad - q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu(x_k,\theta_k)| \\
&\leq \sum_{\substack{x_{k-1},\theta_{k-1},a_{k-1} \\ x_k,\theta_k}} |q^{P_t,\phi_t,\mu_t}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu_t(x_k,\theta_k) + \\
&\qquad - q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu_t(x_k,\theta_k)| + \\
&\quad + \sum_{\substack{x_{k-1},\theta_{k-1},a_{k-1} \\ x_k,\theta_k}} |q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu_t(x_k,\theta_k) + \\
&\qquad - q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})P_t(x_k|x_{k-1},\theta_{k-1},a_{k-1})\mu(x_k,\theta_k)| \\
&\leq \sum_{x_{k-1},\theta_{k-1},a_{k-1}} \left|q^{P_t,\phi_t,\mu_t}(x_{k-1},\theta_{k-1},a_{k-1}) - q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})\right| \\
&\qquad + \sum_{x_k \in X_k} q^{P_t,\phi_t,\mu}(x_k) \sum_{\theta_k \in \Theta} |\mu_t(x_k,\theta_k) - \mu(x_k,\theta_k)| \,.
\end{aligned}$$

Thus, by induction hypothesis, it follows,

$$\sum_{x_k \in X_k} \sum_{\theta_k \in \Theta} \sum_{a_k \in A} \left| q^{P_t,\phi_t,\mu_t}(x_k, \theta_k, a_k) - q^{P_t,\phi_t,\mu}(x_k, \theta_k, a_k) \right| \\ \leq \sum_{s=0}^{k} \sum_{x_s \in X_s} q^{P_t,\phi_t,\mu}(x_s) \|\mu_t(\cdot|x_s) - \mu(\cdot|x_s)\|_1.$$

Bound on $\left|q^{P_t,\phi_t,\mu}(x,\theta,a) - q^{P,\phi_t,\mu}(x,\theta,a)\right|$. To bound this term, we proceed again by induction. Thus, we notice that:

$$\begin{aligned}
&\sum_{x_1 \in X_1} \sum_{\theta_1 \in \Theta} \sum_{a_1 \in A} |q^{P_t,\phi_t,\mu}(x_1, \theta_1, a_1) - q^{P,\phi_t,\mu}(x_1, \theta_1, a_1)| \\
&= \sum_{\theta_0 \in \Theta} \sum_{a_0 \in A} \sum_{x_1 \in X_1} \sum_{\theta_1 \in \Theta} \sum_{a_1 \in A} |\mu(x_0, \theta_0)\phi_t(a_0|x_0, \theta_0) P_t(x_1|x_0, \theta_0, a_0)\mu(x_1, \theta_1)\phi_t(a_1|x_1, \theta_1) \\
&\qquad - \mu(x_0, \theta_0)\phi_t(a_0|x_0, \theta_0) P(x_1|x_0, \theta_0, a_0)\mu(x_1, \theta_1)\phi_t(a_1|x_1, \theta_1)| \\
&= \sum_{\theta_0 \in \Theta} \sum_{a_0 \in A} \mu(x_0, \theta_0)\phi_t(a_0|x_0, \theta_0) \sum_{x_1 \in X_1} |P_t(x_1|x_0, \theta_0, a_0) - P(x_1|x_0, \theta_0, a_0)| \cdot \\
&\qquad \cdot \sum_{\theta_1 \in \Theta} \sum_{a_1 \in A} \mu(x_1, \theta_1)\phi_t(a_1|x_1, \theta_1)| \\
&\leq \sum_{\theta_0 \in \Theta} \sum_{a_0 \in A} q^{P,\phi_t,\mu}(x_0, \theta_0, a_0) \|P_t(\cdot|x_0, \theta_0, a_0) - P(\cdot|x_0, \theta_0, a_0)\|_1.
\end{aligned}$$

Now, we proceed with the induction step,

$$\begin{aligned}
&\sum_{x_k \in X_k} \sum_{\theta_k \in \Theta} \sum_{a_k \in A} |q^{P_t,\phi_t,\mu}(x_k, \theta_k, a_k) - q^{P,\phi_t,\mu}(x_k, \theta_k, a_k)| \\
&= \sum_{\substack{x_{k-1}, \theta_{k-1}, a_{k-1} \\ x_k, \theta_k, a_k}} |q^{P_t,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P_t(x_k|x_{k-1}, \theta_{k-1}, a_{k-1})\mu(x_k, \theta_k)\phi_t(a_k|x_k, \theta_k) + \\
&\qquad - q^{P,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P(x_k|x_{k-1}, \theta_{k-1}, a_{k-1})\mu(x_k, \theta_k)\phi_t(a_k|x_k, \theta_k)| \\
&= \sum_{x_{k-1} \in X_{k-1}} \sum_{\theta_{k-1} \in \Theta} \sum_{a_{k-1} \in A} \sum_{x_k \in X_k} |q^{P_t,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P_t(x_k|x_{k-1}, \theta_{k-1}, a_{k-1}) \\
&\qquad - q^{P,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P(x_k|x_{k-1}, \theta_{k-1}, a_{k-1})| \\
&\leq \sum_{x_{k-1} \in X_{k-1}} \sum_{\theta_{k-1} \in \Theta} \sum_{a_{k-1} \in A} \sum_{x_k \in X_k} |q^{P_t,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P_t(x_k|x_{k-1}, \theta_{k-1}, a_{k-1}) \\
&\qquad - q^{P,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P_t(x_k|x_{k-1}, \theta_{k-1}, a_{k-1})| \\
&+ \sum_{x_{k-1} \in X_{k-1}} \sum_{\theta_{k-1} \in \Theta} \sum_{a_{k-1} \in A} \sum_{x_k \in X_k} |q^{P,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P_t(x_k|x_{k-1}, \theta_{k-1}, a_{k-1}) \\
&\qquad - q^{P,\phi_t,\mu}(x_{k-1}, \theta_{k-1}, a_{k-1}) P(x_k|x_{k-1}, \theta_{k-1}, a_{k-1})|
\end{aligned}$$

$$\leq \sum_{x_{k-1}\in X_{k-1}} \sum_{\theta_{k-1}\in\Theta} \sum_{a_{k-1}\in A} |q^{P_t,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1}) - q^{P,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})|$$

$$+ \sum_{x_{k-1}\in X_{k-1}} \sum_{\theta_{k-1}\in\Theta} \sum_{a_{k-1}\in A} q^{P,\phi_t,\mu}(x_{k-1},\theta_{k-1},a_{k-1})\cdot$$

$$\cdot \|P_t(\cdot|x_{k-1},\theta_{k-1},a_{k-1}) - P(\cdot|x_{k-1},\theta_{k-1},a_{k-1})\|_1.$$

Thus by induction hypothesis we obtain,

$$\sum_{x_k\in X_k} \sum_{\theta_k\in\Theta} \sum_{a_k\in A} |q^{P_t,\phi_t,\mu}(x_k,\theta_k,a_k) - q^{P,\phi_t,\mu}(x_k,\theta_k,a_k)|$$

$$\leq \sum_{s=0}^{k-1} \sum_{x_s\in X_s} \sum_{\theta_s\in\Theta} \sum_{a_s\in A} q^{P,\phi_t,\mu}(x_s,\theta_s,a_s)\|P_t(\cdot|x_s,\theta_s,a_s) - P(\cdot|x_s,\theta_s,a_s)\|_1.$$

Returning to the quantity of interest we have:

$$\sum_{t\in[T]} \|q_t - \widehat{q}_t\|_1 \leq 2 \sum_{t\in[T]} \sum_{k=0}^{L-1} \sum_{s=0}^{k-1} \sum_{x_s\in X_s} \sum_{\theta_s\in\Theta} \sum_{a_s\in A} q^{P,\phi_t,\mu}(x_s,\theta_s,a_s)\|P_t(\cdot|x_s,\theta_s,a_s) +$$

$$- P(\cdot|x_s,\theta_s,a_s)\|_1 + \sum_{t\in[T]} \sum_{k=0}^{L-1} \sum_{s=0}^{k} \sum_{x_s\in X_s} q^{P_t,\phi_t,\mu}(x_s)\|\mu_t(\cdot|x_s) - \mu(\cdot|x_s)\|_1. \tag{5.2}$$

We proceed bounding the first term in Inequality (5.2). Fixing a layer $k \in [0,\dots,L-1]$, employing Azuma-Hoeffding inequality and noticing that

$$\|P_t(\cdot|x_k,\theta_k,a_k) - P(\cdot|x_k,\theta_k,a_k)\|_1 \leq 2,$$

we have, with probability $1-2\delta$:

$$\sum_{t\in[T]} \sum_{s=0}^{k-1} \sum_{x_s,\theta_s,a_s} q^{P,\phi_t,\mu}(x_s,\theta_s,a_s)\|P_t(\cdot|x_s,\theta_s,a_s) - P(\cdot|x_s,\theta_s,a_s)\|_1$$

$$\leq \sum_{s=0}^{k-1} \sum_{t\in[T]} \sum_{x_s,\theta_s,a_s} \sqrt{\frac{2|X_{k(x_s)+1}|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}{\max\{1,N_t(x_s,\theta_s,a_s)\}}}\mathbb{1}_t\{x_s,a_s,\theta_s\} + \sum_{s=0}^{k-1} 2|X_s|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}$$

$$\leq \sum_{s=0}^{k-1} \sqrt{2T|X_s||X_{s+1}|A||\Theta|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)} + \sum_{s=0}^{k-1} 2|X_s|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}$$

$$\leq |X|\sqrt{2T|A||\Theta|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)} + 2|X|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}.$$

Finally summing over L, we have, with probability at least $1-2\delta$ (which derives from a union bound between Azuma-Hoeffding inequality and the bound on

the transitions):

$$\sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k-1}\sum_{x_s\in X_s}\sum_{\theta_s\in\Theta}\sum_{a_s\in A} q^{P,\phi_t,\mu}(x_s,\theta_s,a_s)\|P_t(\cdot|x_s,\theta_s,a_s)-P(\cdot|x_s,\theta_s,a_s)\|_1$$
$$\leq L|X|\sqrt{2T|A||\Theta|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}+2L|X|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}.$$

To bound the remaining term in Inequality (5.2), we proceed as follows,

$$\sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s} q^{P_t,\phi_t,\mu}(x_s)\|\mu_t(\cdot|x_s)-\mu(\cdot|x_s)\|_1$$
$$\leq \sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s}\left(q^{P_t,\phi_t,\mu}(x_s)\pm q^{P,\phi_t,\mu}(x_s)\right)\|\mu_t(\cdot|x_s)-\mu(\cdot|x_s)\|_1$$
$$\leq \sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s} q^{P,\phi_t,\mu}(x_s)\|\mu_t(\cdot|x_s)-\mu(\cdot|x_s)\|_1+2\left(q^{P_t,\phi_t,\mu}(x_s)-q^{P,\phi_t,\mu}(x_s)\right)$$
$$\leq \sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s} q^{P,\phi_t,\mu}(x_s)\|\mu_t(\cdot|x_s)-\mu(\cdot|x_s)\|_1+$$
$$+\sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s}\sum_{\theta_s\in\Theta}\sum_{a_s\in A} 2\left|q^{P_t,\phi_t,\mu}(x_s,\theta_s,a_s)-q^{P,\phi_t,\mu}(x_s,\theta_s,a_s)\right|.$$

The second term is bounded by the previous analysis paying an additional L factor, while, to bound the first terms we apply the Azuma-Hoeffding inequality and proceed as follows:

$$\sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s} q^{P,\phi_t,\mu}(x_s)\sum_{\theta_s\in\Theta}|\mu_t(x_s,\theta_s)-\mu(x_s,\theta_s)|$$
$$\leq \sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s}\mathbb{1}_t\{x_s\}\|\mu_t(\cdot|x_s)-\mu(\cdot|x_s)\|_1+2L|X|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}$$
$$\leq \sum_{t\in[T]}\sum_{k=0}^{L-1}\sum_{s=0}^{k}\sum_{x_s\in X_s}\mathbb{1}_t\{x_s\}\sqrt{\frac{2|\Theta|\ln(T|X|/\delta)}{\max\{1,N_t(x_s)\}}}+2L|X|\sqrt{2T\ln\left(\frac{L}{\delta}\right)}$$
$$\leq 2L\sqrt{2L|X||\Theta|T\ln\left(\frac{T|X|}{\delta}\right)}+2L|X|\sqrt{2T\ln\left(\frac{L}{\delta}\right)},$$

with probability at least $1-2\delta$, given the union bound over the Azuma-Hoeffding and the bound on the outcomes. Finally, with a union bound between the bound on the transitions and the outcomes (which are both encompassed by the clean

event) and the Azuma-Hoeffding inequalities, with probability at least $1 - 4\delta$, we have:

$$\sum_{t\in[T]} \|q_t - \widehat{q}_t\|_1 \leq \mathcal{O}\left(L^2|X|\sqrt{T|A||\Theta|\ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}\right),$$

which concludes the proof. □

Intuitively, Lemma 5.2 is proved by an inductive argument that relates the uncertainty associated with the transition and the prior functions to the $\|\cdot\|_1$-norm difference between q_t and $\widehat{q}_t$ cumulated over the episodes. Lemma 5.2 paves the way for our two main theorems in the full-feedback setting. The first theorem bounds the regret R_T achieved by `OPPS`, while the second one bounds its cumulative violation V_T. Formally:

Theorem 5.1. *Given any $\delta \in (0,1)$, with probability at least $1-7\delta$, Algorithm 5.2 attains*

$$R_T \leq \widetilde{\mathcal{O}}\left(L^2|X|\sqrt{T|A||\Theta|\ln(1/\delta)}\right).$$

Proof. We notice that the regret can be decomposed as follows:

$$\begin{aligned} R_T &= \sum_{t,x,\theta,a} r_S(x,\theta,a)\left(q^*(x,\theta,a) - q_t(x,\theta,a)\right) \\ &= \sum_{t,x,\theta,a} r_S(x,\theta,a)\left(q^*(x,\theta,a) - \widehat{q}_t(x,\theta,a)\right) + r_S(x,\theta,a)\left(\widehat{q}_t(x,\theta,a) - q_t(x,\theta,a)\right). \end{aligned}$$

The second term above is bounded by Holder's inequality and applying Lemma 5.2. To bound the first term we notice that, under the clean event, and according how Algorithm 5.2 works, it holds:

$$\begin{aligned} \sum_{x,\theta,a} r_S(x,\theta,a)\, q^*(x,\theta,a) &\leq \sum_{x,\theta,a} (\overline{r}_{S,t} + \xi_{S,t})(x,\theta,a)\, q^*(x,\theta,a) \\ &\leq \sum_{x,\theta,a} (\overline{r}_{S,t} + \xi_{S,t})(x,\theta,a)\, \widehat{q}_t(x,\theta,a) \\ &\leq \sum_{x,\theta,a} (r_S + 2\xi_{S,t})(x,\theta,a)\, \widehat{q}_t(x,\theta,a). \end{aligned}$$

Notice that the second inequality above holds thanks to the fact that q^* is a feasible solution to `Opt-Opt` thanks to Lemma 5.3. Thus, we have,

$$\begin{aligned} &\sum_{t,x,\theta,a} r_S(x,\theta,a)\left(q^*(x,\theta,a) - \widehat{q}_t(x,\theta,a)\right) \leq 2\sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a)\,\widehat{q}_t(x,\theta,a) \\ &= 2\sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a)\, q_t(x,\theta,a) + 2\sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a)\left((\widehat{q}_t - q_t)(x,\theta,a)\right). \quad (5.3) \end{aligned}$$

The second term above can be bounded using Holder's inequality together with Lemma 5.2, which holds under the clean event with probability at least $1 - 2\delta$. To bound the first term, we apply Lemma 5.4, which also holds under the clean event with probability at least $1 - \delta$. Finally, a union bound completes the proof. □

Theorem 5.2. *Given $\delta \in (0, 1)$, with probability at least $1 - 7\delta$, Algorithm 5.2 attains violation*

$$V_T \leq \widetilde{\mathcal{O}}\left(L^2|X|\sqrt{T|A||\Theta|\ln(1/\delta)}\right).$$

Proof. According to how `Opt-Opt` works, and summing over (x, θ, a), we obtain that for any episode t, under the clean event, the following holds:

$$\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\,(\overline{r}_{R,t} + \xi_{R,t})(x,\theta,a) \;\geq\; \sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\,(\overline{r}_{R,t} - \xi_{R,t})(x,\theta,b^t(a,x)).$$

which, in turn, implies that:

$$\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\,(r_R + 2\xi_{R,t})(x,\theta,a) \;\geq\; \sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\,(r_R - 2\xi_{R,t})(x,\theta,b^t(a,x)).$$

Thus, in the *full-feedback* setting where $\xi_{R,t}(x,\theta,a) = \xi_{R,t}(x,\theta,b^t(a,x))$, we obtain:

$$\begin{aligned}\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\right) &\leq 4\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\xi_{R,t}(x,\theta,a)\\ &\leq 4\sum_{x,\theta} \widehat{q}_t(x,\theta)\xi_{R,t}(x,\theta),\end{aligned}$$

where $\xi_{R,t}(x,\theta) = \sqrt{\frac{\ln(3T|X||\Theta|/\delta)}{\max\{1,N_t(x,\theta)\}}}$. Thus, we have:

$$\begin{aligned}&\sum_{t,x,\theta,a} \widehat{q}_t(x,\theta,a)\Big(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\Big) &(5.4)\\ &\leq 4\sum_{t,x,\theta} \widehat{q}_t(x,\theta)\,\xi_{R,t}(x,\theta)\\ &= 4\sum_{t,x,\theta} q_t(x,\theta)\,\xi_{R,t}(x,\theta) + 4\sum_{t,x,\theta}\left(\widehat{q}_t(x,\theta) - q_t(x,\theta)\right)\xi_{R,t}(x,\theta)\\ &\leq \underbrace{4\sum_{t,x,\theta} q_t(x,\theta)\,\xi_{R,t}(x,\theta)}_{(**)} + \mathcal{O}\left(L^2|X|\sqrt{T|A||\Theta|\ln\frac{T|X||\Theta||A|}{\delta}}\right) &(5.5)\end{aligned}$$

To bound $(**)$, we observe :

$$
\begin{aligned}
(**) &= 4 \sum_{t,x,\theta} q_t(x,\theta)\, \xi_{R,t}(x,\theta) && (5.6)\\
&= 4 \sum_{t,x,\theta} \mathbb{1}_t\{x,\theta\}\, \xi_{R,t}(x,\theta) + 4 \sum_{t,x,\theta} \big(q_t(x,\theta) - \mathbb{1}_t\{x,\theta\}\big)\, \xi_{R,t}(x,\theta) \\
&\leq 4 \sum_{t,x,\theta} \mathbb{1}_t\{x,\theta\}\, \xi_{R,t}(x,\theta) + L\sqrt{2T \ln \frac{1}{\delta}} && (5.7)\\
&\leq \sqrt{9L|X||\Theta|T \ln \frac{3T|X||\Theta|}{\delta}} && (5.8)
\end{aligned}
$$

where Equation (5.5) holds by Holder's inequality and Lemma 5.2, which holds under the clean event, with probability at least $1-2\delta$, Equation (5.7) follows by Azuma-Hoeffding and Equation (5.8) by Cauchy-Schwarz inequality and observing that $1 + \sum_{t\in[T]} \frac{1}{\sqrt{t}} \leq 3\sqrt{T}$.

Combining Equation (5.5) and Equation (5.8), we get:

$$
\sum_{t,x,\theta,a} \widehat{q}_t(x,\theta,a) \Big(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a) \Big) \leq \mathcal{O}\left(L^2|X| \sqrt{T|A||\Theta| \ln \frac{T|X||\Theta||A|}{\delta}} \right)
$$

Finally, returning to the quantity of interest, we bound the cumulative violations as follows:

$$
\begin{aligned}
V_T &:= \sum_{t,x,\theta,a} q_t(x,\theta,a) \left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a) \right) \\
&= \sum_{t,x,\theta,a} \widehat{q}_t(x,\theta,a) \left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a) \right) \\
&\qquad + \sum_{t,x,\theta,a} \left(q_t(x,\theta,a) - \widehat{q}_t(x,\theta,a) \right) \left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a) \right) \\
&\leq \sum_{t,x,\theta,a} \widehat{q}_t(x,\theta,a) \left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a) \right) + \sum_{t,x,\theta,a} \left| q_t(x,\theta,a) - \widehat{q}_t(x,\theta,a) \right| \\
&\leq O\left(L^2|X| \sqrt{T|A||\Theta| \ln \left(\frac{T|X||\Theta||A|}{\delta} \right)} \right),
\end{aligned}
$$

where the last steps hold by Holder's inequality, Lemma 5.2 and the previous bound on the estimated occupancy measure. The final result holds with probability at least $1-7\delta$ employing a union bound over the clean event, which holds with probability at least $1-4\delta$, the Azuma-Hoeffding inequality used above, which holds with probability at least $1-\delta$ and Lemma 5.2, which, under the clean event, holds with probability at least $1-2\delta$. □

In conclusion, in the *full* feedback case, `OPPS` attains R_T and V_T growing as $\widetilde{\mathcal{O}}(\sqrt{T})$. Intuitively, this is made effective by the fact that all the estimators concen-

trate at a $1/\sqrt{T}$ rate. As shown next, achieving such regret and violation bounds is *not* possible anymore under *partial* feedback.

5.3.3 Optimistic optimization problem

In the following section we describe the linear program solved by Algorithm 5.2 and Algorithm 5.3, namely `Opt-Opt`. Intuitively, `Opt-Opt` is the optimistic version of Program (5.1), since the objective is guided by the optimism and the confidence bounds of the estimated parameters are chosen to make constraints easier to be satisfied. Notice that the confidence bounds on the transitions and the prior are applied to the $\|\cdot\|_1$ differences between the empirical and the real mean of the distributions. Thus, in order to insert the aforementioned confidence bounds in a LP-formulation, the related constraints must be linearized by means of additional optimization variables. The linear program solved by Algorithm 5.2 and Algorithm 5.3 is the following.

$$\max_{q,\zeta,\epsilon} \sum_{x\in X_k}\sum_{\theta\in\Theta}\sum_{a\in A}\sum_{x'\in X_{k+1}} q(x,\theta,a,x')\Big(\overline{r}_{S,t}(x,\theta,a)+\xi_{S,t}(x,\theta,a)\Big) \tag{5.9a}$$

$$\sum_{x\in X_k}\sum_{\theta\in\Theta}\sum_{a\in A}\sum_{x'\in X_{k+1}} q(x,\theta,a,x')=1 \qquad \forall k\in[0\dots L-1] \tag{5.9b}$$

$$\sum_{x'\in X_{k-1}}\sum_{\theta\in\Theta}\sum_{a\in A} q(x',\theta,a,x)=\sum_{\theta\in\Theta}\sum_{a\in A}\sum_{x'\in X_{k+1}} q(x,\theta,a,x')$$
$$\forall k\in[0\dots L-1],\ \forall x\in X_k \tag{5.9c}$$

$$q(x,\theta,a,x')-\overline{P}_t(x'|x,\theta,a)\sum_{x''\in X_{k+1}} q(x,\theta,a,x'')\le\epsilon(x,\theta,a,x')$$
$$\forall k\in[0\dots L-1],\ \forall(x,\theta,a,x')\in X_k\times\Theta\times A\times X_{k+1} \tag{5.9d}$$

$$\overline{P}_t(x'|x,a,\theta)\sum_{x''\in X_{k+1}} q(x,\theta,a,x'')-q(x,\theta,a,x')\le\epsilon(x,\theta,a,x')$$
$$\forall k\in[0\dots L-1],\ \forall(x,\theta,a,x')\in X_k\times\Theta\times A\times X_{k+1} \tag{5.9e}$$

$$\sum_{x'\in X_{k+1}}\epsilon(x,\theta,a,x')\le\epsilon_t(x,\theta,a)\sum_{x'\in X_{k+1}} q(x,\theta,a,x')$$
$$\forall k\in[0\dots L-1],\ \forall(x,\theta,a)\in X_k\times\Theta\times A \tag{5.9f}$$

$$q(x,\theta)-\overline{\mu}_t(\theta|x)\sum_{\theta'\in\Theta} q(x,\theta')\le\zeta(x,\theta) \qquad \forall k\in[0\dots L-1],\ \forall(x,\theta)\in X_k\times\Theta \tag{5.9g}$$

$$\overline{\mu}_t(\theta|x)\sum_{\theta'\in\Theta} q(x,\theta')-q(x,\theta)\le\zeta(x,\theta) \qquad \forall k\in[0\dots L-1],\ \forall(x,\theta)\in X_k\times\Theta \tag{5.9h}$$

$$\sum_{\theta\in\Theta}\zeta(x,\theta)\le\zeta_t(x)\sum_{\theta\in\Theta} q(x,\theta) \qquad \forall k\in[0\dots L-1],\ \forall x\in X_k \tag{5.9i}$$

$$\sum_{\substack{\theta\in\Theta\\ x'\in X_{k+1}}} q(x,\theta,a,x')\Big(\overline{r}_{R,t}(x,\theta,a)+\xi_{R,t}(x,\theta,a)-\overline{r}_{R,t}(x,\theta,a')+\xi_{R,t}(x,\theta,a')\Big)\ge 0$$

$$\forall k \in [0 \dots L-1],\ \forall (x,a) \in X_k \times A,\ \forall a' \in A \tag{5.9j}$$
$$q(x,\theta,a,x') \geq 0 \qquad \forall k \in [0 \dots L-1],\ \forall (x,a,x') \in X_k \times \Theta \times A \times X_{k+1} \tag{5.9k}$$

where Objective (5.9a) maximizes the upper confidence bound of the sender reward, Constraint (5.9b) ensures that the occupancy measure sums to 1 for every layer, Constraint (5.9c) is the flow constraint, Constraint (5.9d) is related to the confidence interval on the transition functions, Constraint (5.9e) is still related to the confidence bounds on the transition function, Constraint (5.9f) allows to write linearly the constraints related to the transition functions even if the interval holds for the $\|\cdot\|_1$, Constraint (5.9g) is related to the confidence interval on the outcomes, Constraint (5.9h) is still related to the confidence bounds on the outcomes, Constraint (5.9i) allows to write linearly the constraints related to the outcomes even if the interval holds for the $\|\cdot\|_1$, Constraint (5.9j) is the optimistic constraint for the Incentive Compatibility (IC) property and, finally, Constraint (5.9k) ensures that the occupancy are greater than zero.

We also introduce a lemma that proves that, under the event $\mathcal{E}(\delta)$ every occupancy measure q that is feasible for Problem (5.1), is also feasible for problem Problem (5.9). Formally:

Lemma 5.3. *Given $\delta \in (0,1)$, under event $\mathcal{E}(\delta)$, every $q \in \mathcal{Q}$ that is feasible for Problem (5.1) is also feasible for Problem (5.9) at every round $t \in [T]$.*

Proof. First, observe that under the clean event $\mathcal{E}(\delta)$, both the true transition function P and the prior μ lie within their confidence intervals; hence, they belong to the constrained space defined by `Opt-Opt`. Next, we focus on the incentive compatibility constraints. Let $q^\diamond$ be an incentive-compatible occupancy measure, under $\mathcal{E}(\delta)$, we have:

$$\sum_{\theta\in\Theta, x'\in X_{k+1}} q^\diamond(x,\theta,a,x') \left(\overline{r}_{R,t}(x,\theta,a)+\xi_{R,t}(x,\theta,a)-\overline{r}_{R,t}(x,\theta,\bar{a})+\xi_{R,t}(x,\theta,\bar{a})\right)$$
$$\geq \sum_{\theta\in\Theta, x'\in X_{k+1}} q^\diamond(x,\theta,a,x') \left(r_R(x,\theta,a) - r_R(x,\theta,\bar{a})\right) \geq 0,$$

for any $k \in [L-1], (x,a) \in X_k \times A, \bar{a} \in A$. As a result, if $q^\diamond$ is incentive compatible, it belongs to the optimistic decision space, which concludes the proof. □

5.3.4 Lower bound

We conclude the section by showing that the regret and violation bounds attained by the `OPPS` algorithm with *full* feedback are tight. Formally:

Theorem 5.3. *Let $\eta \in (0, 3/4)$. For every algorithm that guarantees $R_T \leq \mathcal{O}(\sqrt{T})$ with probability at least $1-\eta$, there exists an absolute constant $\psi > 0$ and a problem instance in which the algorithm must have $V_T \geq \Omega(\sqrt{T})$ with probability larger than ψ.*

Proof. We consider two instances with a single possible outcome and a single state. In the following, we omit the dependence on the sender and receiver utility from these parameters. We assume that the sender's utility in the first instance is a deterministic function given by $r_S^1(a_1) = 1$ and $r_S^1(a_2) = 0$, while the receiver's utility is given by $r_R^2(a_1) \sim \text{Be}(1/2 + \epsilon)$ and $r_R^2(a_2) \sim \text{Be}(1/2)$. Meanwhile, the sender's utility in the second instance is $r_S^2(a_1) = 1$ and $r_S^1(a_2) = 0$, while the receiver's utility is equal to $r_R^2(a_1) \sim \text{Be}(1/2 + \epsilon)$ and $r_R^2(a_2) \sim \text{Be}(1/2 + 2\epsilon)$, for some $\epsilon \in (0, 1/2)$. Thus, the sender's regret in the first instance is given by:

$$R_T^1 = \sum_{t=1}^{T} \phi^t(a_2),$$

since the optimal signaling scheme is the one that always recommends $a_1 \in \mathcal{A}$ in the first instance. In the following, we define $\mathbb{P}^1$ (respectively, $\mathbb{P}^2$) as the probability measure in the first (respectively, second) instance. Then, by assumption, we have:

$$\mathbb{P}^1\left(R_T^1 \leq \mathcal{O}(\sqrt{T})\right) = \mathbb{P}^1\left(\sum_{t=1}^{T} \phi^t(a_2) \leq \mathcal{O}(\sqrt{T})\right) \geq 1 - \eta. \tag{5.10}$$

Furthermore, by Pinsker's inequality and Equation (5.10) the following holds

$$\mathbb{P}^2\left(\sum_{t=1}^{T} \phi^t(a_2) \leq \mathcal{O}(\sqrt{T})\right) \geq 1 - \eta - \sqrt{\mathcal{KL}(\mathbb{P}^1, \mathbb{P}^2)}, \tag{5.11}$$

By means of the well known divergence decomposition, we have:

$$\mathcal{KL}(\mathbb{P}^1, \mathbb{P}^2) \leq T \cdot \mathcal{KL}(\text{Be}(1/2 + 2\epsilon), \text{Be}(1/2)) \leq 16\epsilon^2 T, \tag{5.12}$$

where in the latter inequality we used the well known property ensuring that

$$\mathcal{KL}(\text{Be}(p), \text{Be}(q)) \leq \frac{(p-q)^2}{q(1-q)}.$$

Furthermore, by means of the latter inequality and Equation (5.12) we have:

$$\mathbb{P}^2\left(\sum_{t=1}^{T} \phi^t(a_2) \leq \mathcal{O}(\sqrt{T})\right) \geq 1 - \eta - \sqrt{16\epsilon^2 T}$$

We now consider the receiver's violations in the second instance which can be computed as follows:

$$V_T^2 = \sum_{t=1}^{T} \phi^t(a_1)\left(\overline{r}_R^2(a_2) - \overline{r}_R^2(a_1)\right) = \epsilon \sum_{t=1}^{T} \phi^t(a_1).$$

Then, by means of Equation (5.11) we get:

$$\begin{aligned}\mathbb{P}^2\left(V_T^2 \geq \epsilon(T-\mathcal{O}(\sqrt{T}))\right) &= \mathbb{P}^2\left(\epsilon\sum_{t=1}^{T}\phi^t(a_1) \geq \epsilon(T-\mathcal{O}(\sqrt{T}))\right)\\ &= \mathbb{P}^2\left(T-\sum_{t=1}^{T}\phi^t(a_2) \geq T-\mathcal{O}(\sqrt{T})\right)\\ &= \mathbb{P}^2\left(\sum_{t=1}^{T}\phi^t(a_2) \leq \mathcal{O}(\sqrt{T})\right)\\ &\geq 1-\eta-\sqrt{16\epsilon^2 T}.\end{aligned}$$

Finally, by setting $\epsilon = \frac{1}{16\sqrt{T}}$ we get:

$$\mathbb{P}^2\left(V_T^2 \geq \epsilon(T-\mathcal{O}(\sqrt{T}))\right) = \mathbb{P}^2\left(V_T^2 \geq \Omega(\sqrt{T})\right) \geq 1-\eta-\sqrt{16\epsilon^2 T} \geq \frac{3}{4}-\eta,$$

which concludes the proof. □

5.4 The partial feedback case

In this section, we switch the attention to partial feedback. The crucial aspect that makes the case of partial feedback more challenging than the one of full feedback is that, after committing to a signaling policy ϕ_t, the sender does *not* observe sufficient feedback about the persuasiveness of ϕ_t. This makes achieving sublinear violation in the partial feedback case much harder than in the full feedback case.

In order to overcome such a challenge, some episodes of learning must be devoted to the estimation of the quantities involved in persuasiveness constraints. This is necessary to build a suitable approximation of such constraints to be exploited in the remaining episodes, in which an optimistic approach similar to that employed with full feedback must be adopted to control the regret. As a result, there is a trade-off between regret and violation that is determined by the amount of exploration performed. In the rest of this section, we design an algorithm that is able to optimally control such a trade-off.

5.4.1 The `OPPS` algorithm with partial feedback

We extend the `OPPS` algorithm introduced in Section 5.3 to deal with the *partial* feedback case. The idea behind the new algorithm is to split episodes into two phases. The *first* one is an exploration phase with the goal of building a sufficiently-good approximation of persuasiveness constraints, so as to achieve sublinear violation. Such a phase lasts for the first $N|X||\Theta||A|$ episodes, where we let $N := \lceil T^\alpha \rceil$ with $\alpha \in [0,1]$ being a parameter controlling the length of the two phases, given as input to the algorithm. The *second* phase is instead devoted to achieving sublinear regret, and it follows the same steps of `OPPS` with full feedback (Algorithm 5.2). The first phase works by considering each $(x,\theta,a) \in X \times \Theta \times A$ for N episodes. When (x,θ,a) is considered at episode $t \in [T]$, the algorithm commits to a signal-

ing scheme induced by an occupancy measure $\widehat{q}_t$ that maximizes the probability $\sum_{x' \in X} q(x, \theta, a, x')$ of visiting such a triplet, while at the same time satisfying all the constraints of the `Opt-Opt` problem except Equation (5.9j).

Crucially, such a procedure does *not* guarantee that every triplet is visited N times. Indeed, there might be triplets (x, θ, a) that are visited with very low probability.

Algorithm 5.3 Optimistic Persuasive Policy Search (*partial*)

Require: X, Θ, A, T, $\delta \in (0, 1)$, $\alpha \in [0, 1]$
1: $N \leftarrow \lceil T^\alpha \rceil$
2: Initialize all estimators to 0 and all bounds to $+\infty$
3: Initialize counter $C(x, \theta, a)$ to 0 for all (x, θ, a)
4: **for** $t = 1, \dots, T$ **do**
5: Update all estimators $\overline{P}_t, \overline{\mu}_t, \overline{r}_{S,t}, \overline{r}_{R,t}$ and bounds $\epsilon_t, \zeta_t, \xi_{S,t}, \xi_{R,t}$
6: **if** $t \leq N|X||\Theta||A|$ **then**
7: $(x, \theta, a) \leftarrow \arg\min_{(x,\theta,a) \in X \times \Theta \times A} C(x, \theta, a)$
8: $\widehat{q}_t \leftarrow$ Solve `Opt-Opt` with objective $\sum_{x' \in X} q(x, \theta, a, x')$, removing Eq. (5.9j)
9: $C(x, \theta, a) \leftarrow C(x, \theta, a) + 1$
10: **else**
11: $\widehat{q}_t \leftarrow$ Solve `Opt-Opt` (Problem (5.9))
12: **end if**
13: $\phi_t \leftarrow \phi^{\widehat{q}_t}$
14: Run Algorithm 5.1 by committing to ϕ_t
15: Observe *partial* feedback from Algorithm 5.1
16: **end for**

Algorithm 5.3 provides the pseudocode of `OPPS` with *partial* feedback. Notice that the variables $C(x, \theta, a)$ (initialized in Line 3 and updated in Line 9) are counters used to keep track of how many times each triplet (x, θ, a) is considered during the first phase, namely when $t \leq N|X||\Theta||A|$.

Moreover, the algorithm ensures that every triplet is considered exactly N times during the first phase, by selecting them accordingly as in Line 7. Let us also observe that Algorithm 5.3 updates all the estimators and bounds (by using partial feedback) and selects the signaling policy ϕ_t as done by Algorithm 5.2. The main difference with respect to Algorithm 5.2 is that $\widehat{q}_t$ used to define ϕ_t is computed in a different way during the first (exploration) phase (see Line 8).

5.4.2 Algorithm analysis with partial feedback

In the following, we prove the guarantees attained by `OPPS` with *partial* feedback. We start by stating the following result on the regret attained by the algorithm.

Lemma 5.4. *Under the event $\mathcal{E}(\delta)$, with probability at least $1 - \delta$, it holds:*

$$\sum_{t \in [T]} \sum_{x \in X, \theta \in \Theta, a \in A} \xi_{S,t}(x, \theta, a) q_t(x, \theta, a) \leq \mathcal{O}\left(\sqrt{L|X||\Theta||A|T \ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}\right)$$

$$\sum_{t \in [T]} \sum_{x \in X, \theta \in \Theta, a \in A} \xi_{R,t}(x, \theta, a) q_t(x, \theta, a) \leq \mathcal{O}\left(\sqrt{L|X||\Theta||A|T \ln\left(\frac{T|X||\Theta||A|}{\delta}\right)}\right)$$

Proof. For the sake of presentation, we provide the proof of the upper bound only for the term related to the sender, as the term regarding the receiver can be bounded using the same analysis.

$$\sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a) q_t(x,\theta,a) \leq \sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a) \mathbb{1}_t\{x,\theta,a\} + L\sqrt{2T \ln \frac{1}{\delta}} \tag{5.13}$$

$$= \sum_{t,x,\theta,a} \sqrt{\frac{\ln(3T|X||\Theta||A|/\delta)}{\max\{1, N_t(x,\theta,a)\}}} \mathbb{1}_t\{x,\theta,a\} + L\sqrt{2T \ln \frac{1}{\delta}}$$

$$\leq \sqrt{9 \ln\left(\frac{3T|X||\Theta||A|}{\delta}\right)} \sum_{x,\theta,a} \sqrt{N_T(x,\theta,a)} + L\sqrt{2T \ln \frac{1}{\delta}} \tag{5.14}$$

$$\leq \sqrt{9 \ln\left(\frac{3T|X||\Theta||A|}{\delta}\right)} \sqrt{|X||\Theta||A| \sum_{x,\theta,a} N_T(x,\theta,a)} + L\sqrt{2T \ln \frac{1}{\delta}} \tag{5.15}$$

$$\leq \sqrt{9L|X||\Theta||A|T \ln\left(\frac{3T|X||\Theta||A|}{\delta}\right)} + L\sqrt{2T \ln \frac{1}{\delta}}, \tag{5.16}$$

where Equation (5.13) holds by the Azuma–Hoeffding inequality with probability $1-\delta$, Equation (5.14) follows from the bound $1+\sum_{t\in[T]} 1/\sqrt{t} \leq 3\sqrt{T}$, Equation (5.15) follows from the Cauchy–Schwarz inequality, and Equation (5.16) holds since

$$\sum_{x,\theta,a} N_T(x,\theta,a) \leq LT,$$

which concludes the proof. □

Theorem 5.4. *Given any $\delta \in (0,1)$, with probability at least $1-7\delta$, Algorithm 5.3 attains regret*

$$R_T \leq \widetilde{\mathcal{O}}\left(NL|X||\Theta||A| + L^2|X|\sqrt{T|A||\Theta| \ln(1/\delta)}\right).$$

Proof. As a first step, we decompose the sender's regret as follows.

$$R_T = \sum_{t,x,\theta,a} r_S(x,\theta,a)\left(q^*(x,\theta,a) - q_t(x,\theta,a)\right)$$

$$= \sum_{t,x,\theta,a} r_S(x,\theta,a)\big((q^* - \widehat{q}_t)(x,\theta,a)\big) + \sum_{t,x,\theta,a} r_S(x,\theta,a)\big((\widehat{q}_t - q_t)(x,\theta,a)\big)$$

$$\leq \sum_{t,x,\theta,a} r_S(x,\theta,a)\left((q^* - \widehat{q}_t)(x,\theta,a)\right) + \mathcal{O}\left(L^2|X|\sqrt{T|A||\Theta| \ln\left(\tfrac{T|X||\Theta||A|}{\delta}\right)}\right) \tag{5.17}$$

We observe that the last inequality holds under the event $\mathcal{E}(\delta)$, with probability at least $1-2\delta$, and it is derived by applying Holder's inequality and Lemma 5.2.

To bound the first term in Equation (5.17), notice that under $\mathcal{E}(\delta)$:

$$\begin{aligned}\sum_{x,\theta,a} r_S(x,\theta,a)\, q^*(x,\theta,a) &\leq \sum_{x,\theta,a} (\overline{r}_{S,t} + \xi_{S,t})(x,\theta,a)\, q^*(x,\theta,a) \\ &\leq \sum_{x,\theta,a} (r_S + 2\xi_{S,t})(x,\theta,a)\, \widehat{q}_t(x,\theta,a)\end{aligned}$$

for each $t > N|X||\Theta||A|$, by the optimality of $\widehat{q}_t$. Hence, defining $t_0 \coloneqq N|X||\Theta||A|$, we have

$$\begin{aligned}&\sum_{t,x,\theta,a} r_S(x,\theta,a)\left((q^* - \widehat{q}_t)(x,\theta,a)\right) \\ &= \sum_{t\leq t_0}\sum_{x,\theta,a} r_S(x,\theta,a)\left((q^* - \widehat{q}_t)(x,\theta,a)\right) + \sum_{t>t_0}\sum_{x,\theta,a} r_S(x,\theta,a)\left((q^* - \widehat{q}_t)(x,\theta,a)\right) \\ &\leq NL|X||\Theta||A| + 2\sum_{t,x,\theta,a} \xi_{S,t}(x,\theta,a)\left(\widehat{q}_t(x,\theta,a) \pm q_t(x,\theta,a)\right) \\ &\leq NL|X||\Theta||A| + \mathcal{O}\left(L^2|X|\sqrt{T|A||\Theta|\ln\left(\tfrac{T|X||\Theta||A|}{\delta}\right)}\right).\end{aligned}$$

Substituting this into Equation (5.17), we obtain

$$R_T \leq \mathcal{O}\left(NL|X||\Theta||A| + L^2|X|\sqrt{T|A||\Theta|\ln\left(\tfrac{T|X||\Theta||A|}{\delta}\right)}\right),$$

which holds with probability at least $1 - 7\delta$, by a union bound and the fact that $\mathcal{E}(\delta)$ occurs with probability at least $1 - 4\delta$. □

In order to prove Theorem 5.4, we split the analysis into two cases: one targets exploration episodes in the first phase of the algorithm, while the other is concerned with the subsequent (exploitation) phase. In the first N episodes in which the `OPPS` algorithm explores without being driven by the `Opt-Opt` objective, the algorithm incurs in linear regret. Instead, in the second phase, `OPPS` employs an optimistic approach, since the algorithm is driven by the objective of the `Opt-Opt` problem. Thus, in the second phase, the algorithm attains regret sublinear in T. The two cases combined give the regret bound provided in Theorem 5.4. Next, we state the result on the violations attained by `OPPS`.

Lemma 5.5. *Under the event $E(\delta)$, with probability at least $1 - 3\delta$, it holds*

$$V_{\text{opt}} \leq K_\delta + 2\sum_{t>N\ell}\sum_{x,\theta,a} q_t(x,\theta,a)\xi_{R,t}(x,\theta,b^t(a,x)),$$

where $\ell := |X||\Theta||A|$,

$$V_{\text{opt}} := \sum_{t>N\ell}\sum_{x,\theta,a} q_t(x,\theta,a)\big(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\big),$$

and $K_\delta = O\left(L^2|X|\sqrt{T|A||\Theta|\log\left(\tfrac{T|X||\Theta||A|}{\delta}\right)}\right).$

Proof. As a first step, we observe that by employing the definition of $\xi_{R,t}$ and noticing that $\widehat{q}_t$ is a feasible solution to LP (5.9) for each $t > N|X||\Theta||A|$ under the event $\mathcal{E}(\delta)$, we have:

$$\begin{aligned}&\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\left((r_R + 2\xi_{R,t})(x,\theta,a) - (r_R - 2\xi_{R,t})(x,\theta,b^t(a,x))\right)\\&\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\left((\overline{r}_{R,t} + \xi_{R,t})(x,\theta,a) - (\overline{r}_{R,t} - \xi_{R,t})(x,\theta,b^t(a,x))\right) \geq 0.\end{aligned} \tag{5.18}$$

Then, rearranging the above inequality we get:

$$\begin{aligned}&\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\right)\\&\qquad\leq 2\sum_{x,\theta,a} \widehat{q}_t(x,\theta,a)\left(\xi_{R,t}(x,\theta,a) + \xi_{R,t}(x,\theta,b^t(a,x))\right).\end{aligned} \tag{5.19}$$

Furthermore, we can decompose the receivers' violations as follows:

$$\begin{aligned}V_{\text{opt}} &= \sum_{t,x,\theta,a} (q_t(x,\theta,a) \pm \widehat{q}_t(x,\theta,a))\left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\right)\\&\leq K_\delta + \sum_{t,x,\theta,a} \widehat{q}_t(x,\theta,a)\left(r_R(x,\theta,b^t(a,x)) - r_R(x,\theta,a)\right)\\&\leq K_\delta + 2\sum_{t,x,\theta,a} (\widehat{q}_t(x,\theta,a) \pm q_t(x,\theta,a))\left(\xi_{R,t}(x,\theta,a) + \xi_{R,t}(x,\theta,b^t(a,x))\right)\\&\leq K_\delta + \sum_{t,x,\theta,a} q_t(x,\theta,a)\left(2\xi_{R,t}(x,\theta,a) + 2\xi_{R,t}(x,\theta,b^t(a,x))\right)\\&\leq K_\delta + 2\sum_{t,x,\theta,a} q_t(x,\theta,a)\xi_{R,t}(x,\theta,b^t(a,x)).\end{aligned}$$

The first and third inequalities above hold by Lemma 5.2, the second inequality is a consequence of Equation (5.19), and the third inequality follows by means of Lemma 5.4, which holds with a probability of at least $1-\delta$. Therefore, employing a union bound over the events of Lemma 5.2 and Lemma 5.4, the previous result holds with probability at least $1 - 3\delta$, under the clean event. □

Lemma 5.6. *Fix an episode $t > N\ell$, a triplet $(x,\theta,a) \in X \times \Theta \times A$, and let $a' := b^t(a,x)$. For every $i \in [N]$, let $\tau_i(x,\theta,a')$ be the i-th exploration episode in which Algorithm 5.3 targets the triplet (x,θ,a'). Let $\widehat{q}_{\tau_i(x,\theta,a')}$ be the occupancy measure returned by the optimization problem solved at that line. Under the clean event, it holds that*

$$q_t(x,\theta,a) \leq \widehat{q}_{\tau_i(x,\theta,a')}(x,\theta,a') \qquad \forall i \in [N].$$

Proof. Fix $t > N\ell$, $(x, \theta, a) \in X \times \Theta \times A$, and let $a' := b^t(a, x)$. Consider the policy executed at episode t. We construct another policy as follows. The new policy behaves exactly as the policy used at episode t up to state x. Whenever state x is reached and the original policy recommends action a, the new policy recommends action a' instead. The continuation after this recommendation is chosen so as to induce a valid occupancy measure under the true transition function.

The probability of reaching (x, θ) before the recommendation is made is unchanged. Hence, if $\widetilde{q}$ denotes the true occupancy measure induced by the new policy, then

$$\widetilde{q}(x, \theta, a') \geq q_t(x, \theta, a).$$

Moreover, $\widetilde{q}$ is a valid occupancy measure for the true prior and the true transition function. Under the clean event, the true prior and the true transition function belong to the confidence sets used in the optimization problem. Therefore, $\widetilde{q}$ satisfies all constraints of `Opt-Opt` in the first phase.

The only constraint that $\widetilde{q}$ might violate is the optimistic incentive compatibility constraint (5.9j). However, this constraint is not imposed in the first phase. Therefore, $\widetilde{q}$ is feasible for the optimization problem solved at episode $\tau_i(x, \theta, a')$ when the targeted triplet is (x, θ, a').

Since the objective of that optimization problem is precisely to maximize the occupancy of (x, θ, a'), we get

$$\widehat{q}_{\tau_i(x,\theta,a')}(x, \theta, a') \geq \widetilde{q}(x, \theta, a') \geq q_t(x, \theta, a),$$

which concludes the proof. □

Theorem 5.5. *Let $\delta \in (0, 1)$ and let Algorithm 5.3 be run with the modified exploration phase described above. Then, with probability at least $1 - \delta$, the cumulative violation satisfies*

$$V_T \leq \widetilde{O}\left(L^2|A|(|X||\Theta||A|)^2 \frac{T}{\sqrt{N}} + LN|X||\Theta||A| + K_\delta \right),$$

where $K_\delta = O\left(L^2|X|\sqrt{T|A||\Theta| \log\left(\frac{T|X||\Theta||A|}{\delta}\right)} \right)$.

Proof. We split the cumulative violation into the contribution of the exploration episodes and that of the exploitation episodes: $V_T = V_{\text{exp}} + V_{\text{opt}}$, where

$$V_{\text{exp}} := \sum_{t=1}^{N\ell} \sum_{x,\theta,a} q_t(x, \theta, a) \left(r_R(x, \theta, b^t(a, x)) - r_R(x, \theta, a) \right),$$

and

$$V_{\text{opt}} := \sum_{t=N\ell+1}^{T} \sum_{x,\theta,a} q_t(x, \theta, a) \left(r_R(x, \theta, b^t(a, x)) - r_R(x, \theta, a) \right).$$

We first bound the exploration term. Since rewards are in $[0, 1]$ and the total

mass of the occupancy measure in each episode is at most L, every exploration episode contributes at most L to the violation. Therefore, $V_{\text{exp}} \leq LN\ell$.

We focus on the exploitation term. Under the clean event and the event of Lemma 5.2, by Lemma 5.5 we have

$$V_{\text{opt}} \leq K_\delta + 2 \sum_{t=N\ell+1}^{T} \sum_{x,\theta,a} q_t(x,\theta,a)\xi_{R,t}(x,\theta,b^t(a,x)),$$

where

$$K_\delta = O\left(L^2|X|\sqrt{T|A||\Theta|\log\left(\frac{T|X||\Theta||A|}{\delta}\right)}\right).$$

Let

$$S_{\text{opt}} := \sum_{t=N\ell+1}^{T} \sum_{x,\theta,a} q_t(x,\theta,a)\xi_{R,t}(x,\theta,b^t(a,x)).$$

We bound S_{opt}. For every $z = (x,\theta,a) \in X \times \Theta \times A$, define

$$\widehat{Q}_N(z) := \sum_{i=1}^{N} \widehat{q}_{\tau_i(z)}(z),$$

where $\tau_i(z)$ is the exploration episode in epoch i targeting triplet z. Moreover, let

$$\overline{Q}_N(z) := \sum_{i=1}^{N} q_{\tau_i(z)}(z),$$

where $q_{\tau_i(z)}$ is the true occupancy measure induced by the policy executed at episode $\tau_i(z)$. Finally, let

$$M_N(z) := \sum_{i=1}^{N} \mathbf{1}_{\tau_i(z)}\{z\}$$

be the number of times triplet z is actually observed in the exploration episodes targeting z. Fix $t > N\ell$ and (x,θ,a). Let $a' := b^t(a,x)$ and $z' := (x,\theta,a')$. By Lemma 5.6, for every $i \in [N]$,

$$q_t(x,\theta,a) \leq \widehat{q}_{\tau_i(z')}(z').$$

Averaging over $i \in [N]$, we obtain

$$q_t(x,\theta,a) \leq \frac{\widehat{Q}_N(z')}{N}.$$

Moreover, since all exploration samples have already been collected before an episode t, we have $N_t(z') \geq M_N(z')$. Therefore,

$$\xi_{R,t}(z') \leq \frac{C_\delta}{\sqrt{\max\{1, M_N(z')\}}},$$

where C_δ hides the logarithmic confidence factor. Hence,

$$\begin{aligned} S_{opt} &\leq C_\delta \sum_{t=N\ell+1}^{T} \sum_{x,\theta,a} \frac{\widehat{Q}_N(x,\theta,b^t(a,x))}{N\sqrt{\max\{1, M_N(x,\theta,b^t(a,x))\}}} \\ &\leq C_\delta |A| \sum_{t=N\ell+1}^{T} \sum_{z \in X\times\Theta\times A} \frac{\widehat{Q}_N(z)}{N\sqrt{\max\{1, M_N(z)\}}} \\ &\leq C_\delta |A| T \sum_{z \in X\times\Theta\times A} \frac{\widehat{Q}_N(z)}{N\sqrt{\max\{1, M_N(z)\}}}. \end{aligned}$$

We now control the last sum. The sequence

$$\mathbf{1}_{\tau_i(z)}\{z\} - q_{\tau_i(z)}(z), \qquad i \in [N],$$

is a martingale difference sequence bounded in $[-1, 1]$. Thus, using the Azuma–Hoeffding and a union bound over all triplets, with probability at least $1-\delta$,

$$\overline{Q}_N(z) \leq M_N(z) + \sqrt{2N \log\left(\frac{|X||\Theta||A|}{\delta}\right)} \qquad \forall z \in X \times \Theta \times A.$$

For every triplet z, define

$$D_N(z) := \sum_{i=1}^{N} \left|\widehat{q}_{\tau_i(z)}(z) - q_{\tau_i(z)}(z)\right|.$$

Then

$$\widehat{Q}_N(z) \leq \overline{Q}_N(z) + D_N(z).$$

Moreover, by Lemma 5.2 applied to the exploration episodes,

$$\sum_{z \in X\times\Theta\times A} D_N(z) \leq \sum_{t=1}^{N\ell} \|\widehat{q}_t - q_t\|_1 \leq \widetilde{O}\left(L^2 |X| \sqrt{N\ell |A||\Theta|}\right).$$

Therefore,

$$\begin{aligned}
&\sum_z \frac{\widehat{Q}_N(z)}{N\sqrt{\max\{1, M_N(z)\}}} \\
&\leq \sum_z \frac{M_N(z)}{N\sqrt{\max\{1, M_N(z)\}}} + \sum_z \frac{\sqrt{2N\log\left(\frac{|X||\Theta||A|}{\delta}\right)}}{N\sqrt{\max\{1, M_N(z)\}}} + \frac{1}{N}\sum_z D_N(z) \\
&\leq \widetilde{O}\left(\frac{|X||\Theta||A|}{\sqrt{N}}\right) + \widetilde{O}\left(\frac{L^2|X|\sqrt{\ell|A||\Theta|}}{\sqrt{N}}\right).
\end{aligned}$$

In the last inequality we used $M_N(z) \leq N$ for every z. Plugging this bound into the previous display gives

$$S_{\text{opt}} \leq \widetilde{O}\left(|A|T\left(\frac{|X||\Theta||A|}{\sqrt{N}} + \frac{L^2|X|\sqrt{\ell|A||\Theta|}}{\sqrt{N}}\right)\right).$$

In particular, this can be written as

$$S_{\text{opt}} \leq \widetilde{O}\left(L^2|A|(|X||\Theta||A|)^2\frac{T}{\sqrt{N}}\right),$$

Combining the bounds on V_{exp}, V_{opt}, and S_{opt}, we obtain

$$V_T \leq \widetilde{O}\left(L^2|A|(|X||\Theta||A|)^2\frac{T}{\sqrt{N}} + LN|X||\Theta||A| + K_\delta\right),$$

as claimed. □

Proving Theorem 5.5 requires a non-trivial analysis. The main difficulty is to control the violation accumulated after the exploration phase, when the algorithm starts solving the original `Opt-Opt` problem. During exploration, the algorithm solves modified optimization problems in which the objective is chosen so as to maximize the occupancy of a target triplet (x, θ, a), and the optimistic IC constraints are not imposed. Thus, the violation accumulated during this phase is bounded directly by the number of exploration episodes.

For the subsequent episodes, the analysis upper bounds the occupancy measure of each triplet (x, θ, a) by the occupancy measure obtained in a previous exploration episode targeting the same triplet, relative to the best response of the receiver in state x upon receiving recommendation a. This allows us to relate the violation accumulated during exploitation to the number of samples collected during exploration. Theorems 5.4 and 5.5 establish the trade-off between regret and violation achieved by the `OPPS` algorithm. Indeed, by recalling the definition of N (see Line 1 in Algorithm 5.3), it is easy to see that the algorithm attains $R_T \leq \widetilde{\mathcal{O}}(T^\alpha)$ and $V_T \leq \widetilde{\mathcal{O}}\left(T^{\max\{\alpha, 1-\alpha/2\}}\right)$, where $\alpha \in [1/2, 1]$ is the parameter controlling the trade-off, given as input to the algorithm.

5.4.3 Lower bound

We conclude the section and the chapter by providing a lower bound for our setting. Its main idea is to consider two instances of episodic MPP involving a receiver with two actions a_1, a_2 such that only a_1 provides positive reward to the sender. In one instance, receiver's rewards by playing a_1 are higher than those obtained by taking a_2, while in the second instance the opposite holds. As a result, recommending action a_1 results in low regret in the first instance and high violation in the second one, while recommending action a_2 results in low violation in the second instance and high regret in the first one. This leads to the trade-off formally stated by the following theorem.

Theorem 5.6. *Given $\alpha \in [1/2, 1]$ and $T \geq 1$, there is no algorithm achieving both $R_T = o(T^{\alpha})$ and $V_T = o(T^{1-\alpha/2})$ with probability greater or equal to a constant ψ.*

Proof. We consider two instances with a single possible outcome and a single state. In the following, we omit the dependence on the sender and receiver utility from these parameters. We assume that the sender's utility in the first instance is a deterministic function given by $r_S^1(a_1) = 1$ and $r_S^1(a_2) = 0$, while the receiver's utility is given by $r_R^2(a_1) \sim \text{Be}(1/2 + \epsilon)$ and $r_R^2(a_2) \sim \text{Be}(1/2)$. Meanwhile, the sender's utility in the second instance is $r_S^2(a_1) = 1$ and $r_S^1(a_2) = 0$, while the follower's utility is equal to $r_R^2(a_1) \sim \text{Be}(1/2 + \epsilon)$ and $r_R^2(a_2) \sim \text{Be}(1/2 + 2\epsilon)$, for some $\epsilon \in (0, 1/2)$. Thus, the sender's regret in the first instance is given by:

$$R_T^1 = \sum_{t=1}^{T} \phi^t(a_2),$$

since the optimal signaling scheme is the one that always recommends action $a_1 \in \mathcal{A}$ in the first instance. In the following, we define $\mathbb{P}^1$ (respectively, $\mathbb{P}^2$) as the probability measure induced by recommending, at each round, signaling schemes according to some algorithm in the first (respectively, second) instance. Then, we bound the probability that the regret in the first instance is larger than a constant $C \in \mathbb{N}$ as follows:

$$\mathbb{P}^1\left(R_T^1 \leq C\right) = \mathbb{P}^1\left(\sum_{t=1}^{T} \phi^t(a_2) \leq C\right) \geq 1 - \eta, \tag{5.20}$$

for some $\eta \in (0, 1)$. Furthermore, by Pinsker's inequality and Equation (5.20) the following holds.

$$\mathbb{P}^2\left(\sum_{t=1}^{T} \phi^t(a_2) \leq C\right) \geq 1 - \eta - \sqrt{\mathcal{KL}(\mathbb{P}^1, \mathbb{P}^2)}, \tag{5.21}$$

where we denote with $\mathcal{KL}(\cdot, \cdot)$ the Kullback-Leibler divergence between two probability measure. By means of the well known divergence decomposition,

we have:

$$\mathcal{KL}(\mathbb{P}^1, \mathbb{P}^2) \leq \mathbb{E}^1\left[\sum_{t=1}^{T} \phi^t(a_2)\right] \mathcal{KL}(\text{Be}(1/2 + 2\epsilon), \text{Be}(1/2))$$
$$\leq 16\epsilon^2 \mathbb{E}^1\left[\sum_{t=1}^{T} \phi^t(a_2)\right], \tag{5.22}$$

where in the latter inequality we used the well known property ensuring that

$$\mathcal{KL}(\text{Be}(p), \text{Be}(q)) \leq \frac{(p-q)^2}{q(1-q)}.$$

Then, by reverse Markov inequality and Equation (5.20) we get:

$$\mathbb{E}^1\left[\sum_{t=1}^{T} \phi^t(a_2)\right] \leq \mathbb{P}^1\left(\sum_{t=1}^{T} \phi^t(a_2) \geq C\right)(T-C) + C \leq \eta(T-C) + C,$$

Furthermore, by means of the latter inequality and Equation (5.22) we have:

$$\mathbb{P}^2\left(\sum_{t=1}^{T} \phi^t(a_2) \leq C\right) \geq 1 - \eta - \sqrt{16\epsilon^2(\eta(T-C)+C)}$$

We now consider the receiver's violations in the second instance which can be computed as follows:

$$V_T^2 = \sum_{t=1}^{T} \phi^t(a_1)\left(\bar{r}_R^2(a_2) - \bar{r}_R^2(a_1)\right) = \epsilon \sum_{t=1}^{T} \phi^t(a_1).$$

Then, by means of Equation (5.21) we get:

$$\begin{aligned}
\mathbb{P}^2\left(V_T^2 \geq \epsilon(T-C)\right) &= \mathbb{P}^2\left(\epsilon \sum_{t=1}^{T} \phi^t(a_1) \geq \epsilon(T-C)\right) \\
&= \mathbb{P}^2\left(T - \sum_{t=1}^{T} \phi^t(a_2) \geq T - C\right) \\
&= \mathbb{P}^2\left(\sum_{t=1}^{T} \phi^t(a_2) \leq C\right) \\
&\geq 1 - \eta - \sqrt{16\epsilon^2(\eta(T-C)+C)}.
\end{aligned}$$

Finally, by setting $C = \frac{T^\alpha}{2}$ and $\epsilon = \frac{T^{-\alpha/2}}{16}$ and $\eta = \frac{T^{\alpha-1}}{2}$ we get:

$$\begin{aligned}
\mathbb{P}^1\left(R_T^1 \leq C\right) &= \mathbb{P}^1\left(R_T^1 \leq \frac{T^\alpha}{2}\right) \\
&\geq 1 - \eta
\end{aligned}$$

$$\geq 1 - \frac{T^{\alpha-1}}{2} \geq \frac{1}{2},$$

since $\alpha \in [1/2, 1]$. Then, the latter result implies that:

$$\begin{aligned}
\mathbb{P}^2\left(V_T^2 \geq \frac{1}{32}T^{1-\alpha/2}\right) &\geq \mathbb{P}^2\left(V_T^2 \geq \epsilon(T-C)\right) \\
&\geq 1 - \eta - \sqrt{16\epsilon^2(\eta(T-C)+C)} \\
&\geq 1 - \frac{T^{\alpha-1}}{2} - \sqrt{\frac{T^{-\alpha}}{16}\left(\frac{T^\alpha}{2} + \frac{T^\alpha}{2}\right)} \geq \frac{1}{4},
\end{aligned}$$

which concludes the proof. □

5.5 Related works

The work that is most related to this chapter is [Wu et al., 2022], which introduces MPPs. Specifically, Wu et al. [2022] study settings where the sender knows everything about receivers' rewards, with the only elements unknown to them being their rewards, transition probabilities, and prior distributions over outcomes. Moreover, they also assume that the receivers know everything they need about the environment, so as to select a best-response action, and that all rewards are deterministic. In contrast, we consider MPP settings in which sender and receivers have no knowledge of the environment, including their rewards, which we assume to be stochastic. Moreover, Wu et al. [2022] obtain a regret bound of the order of $\widetilde{\mathcal{O}}(\sqrt{T}/D)$, where D is a parameter related to receivers' rewards. Notice that such a dependence is particularly unpleasant, as D may be exponentially large in instances in which there are some receivers' actions that are best responses only for a "small" space of information-disclosure policies. Other works related to ours are [Gan et al., 2022b], which studies a Bayesian persuasion problem where a sender sequentially interacts with a myopic receiver in a multi-state environment, and [Bernasconi et al., 2023b, Gan et al., 2023b], which addresses MPPs with a farsighted receiver. These two works considerably depart from ours, as they both assume that the sender knows everything about the environment, including transitions, priors, and rewards. Thus, they are *not* concerned with learning problems, but with the problem of computing optimal information-disclosure policies. Finally, [Bernasconi et al., 2022] studies settings where a sender faces a farsighted receiver in a sequential environment with a tree structure, addressing the case in which the only elements unknown to the sender are the prior distributions over outcomes, while rewards are deterministic and known. The tree structure considerably eases the learning task, as it allows to express sender's expected rewards linearly in variables defining information-disclosure policies. Intuitively, this allows to factor the uncertainty about transitions in the rewards at the leaves of the tree.

Part III

Learning in Contract Design

CHAPTER 6

Learning bounded contracts

In this chapter, we present an algorithm for learning "approximately-optimal" contracts, requiring a number of principal-agent interactions polynomial in the size of the instance when the number of agent's actions n is constant.

We first characterize the learning task faced by the principal and formalize the problem of learning bounded contracts. Then we describe our algorithm, named `Discover-and-Cover`. Our high-level approach is inspired by techniques developed for learning optimal strategies in Stackelberg games presented in Chapter 3. However, contract design problems introduce several technical challenges that require the adoption of new techniques. In particular, after each round the principal only observes an outcome drawn from the distribution induced by the agent's best response, rather than the action itself. To address this, we design a new procedure whose goal is to group together agent actions associated with "similar" outcome distributions and to construct a set of *meta-actions*, where each meta-action represents one or more (unknown) agent actions. Once a set of meta-actions has been identified, our approach attempts to cover the contract space by means of suitable approximate best-response regions associated with these meta-actions. This leads to a trial-and-error strategy, in which the covering procedure is restarted whenever the set of meta-actions must be updated.

Finally, we prove that our algorithm guarantees to return a contract that is at most $\rho > 0$ suboptimal with respect to the optimal bounded one within a number of rounds proportional to $\widetilde{\mathcal{O}}(m^n/\rho^4 \cdot \log(1/\delta))$ with probability at least $1 - \delta$, and we also show how this approach can be suitably modified to obtain a $\widetilde{\mathcal{O}}(m^n \cdot T^{4/5})$ no-regret algorithm.

6.1 Learning optimal bounded contracts

In this chapter, we focus on settings where the principal and the agent interact over multiple rounds, each corresponding to a repetition of the same hidden-action principal-agent problem. The principal has no prior knowledge of the agent and aims to learn an optimal contract. In each round, the principal-agent interaction proceeds as follows:

(i) The principal commits to a contract $p \in \mathbb{R}_+^m$.

(ii) The agent plays action $a^\star(p)$, which is *not* observed by the principal.

(iii) The principal observes the outcome $\omega \sim F_{a^\star(p)}$ realized by the agent's action.

The principal only knows the set Ω of possible outcomes and their associated rewards r_ω, while they do *not* know anything about agent's actions $\mathcal{A}$, their associated distributions F_a, and their costs c_a.

The goal is to design algorithms for the principal that prescribe how to select a contract at each round in order to learn an (approximately) optimal contract. Ideally, we would like an algorithm that, given an additive approximation error $\rho > 0$ and a probability $\delta \in (0, 1)$, is guaranteed to identify a ρ-optimal contract with probability at least $1 - \delta$ by using the minimum possible number of rounds.[1] The number of rounds required by such an algorithm can be regarded as the *sample complexity* of learning optimal contracts in hidden-action principal-agent problems [Zhu et al., 2023]. However, Theorem 6.1 establishes a negative result for this task. Intuitively, this is because there exist instances in which an approximately optimal contract prescribes a large payment on some outcome. Thus, for any algorithm and any number of rounds N, the payments used up to round N may *not* be large enough to learn such an approximately optimal contract.

Theorem 6.1. *For any number of rounds $N \in \mathbb{N}$, there is no algorithm that is guaranteed to find a κ-optimal contract with probability greater than or equal to $1 - \delta$ by using less than N rounds, where $\kappa, \delta > 0$ are some suitable absolute constants.*

Proof. We consider a family of instances parametrized by $\epsilon \in (0, 1/80)$. Each instance has action set $\mathcal{A} = \{a_1, a_2\}$ and outcome set $\Omega = \{\omega_1, \omega_2, \omega_3\}$. Furthermore, the distributions over the outcomes of the two actions are defined as follows: $F_{a_1} = (1/2, 0, 1/2)$ and $F^\epsilon_{a_2} = (0, \epsilon, 1 - \epsilon)$ with associated cost of $c_{a_1} = 0$ and $c_{a_2} = 1/4$, respectively. In all the instances the principal's reward is equal to $r = (0, 0, 1)$, while the optimal contract is equal to $p^* = (0, 1/4\epsilon, 0)$, resulting in a principal's expected utility of $u(p^*) = 3/4 - \epsilon$.

As a first step, we show that if $p_{\omega_2} \leq 1/8\epsilon$, then the principal's utility is at most $9/80$-optimal. To show that, we consider two cases.

- $p_{\omega_2} \leq 1/8\epsilon$ and $p \in \mathcal{P}_{a_1}$. In this case, the highest expected utility achieved by the principal is at most $1/2$, which occurs when they commit to the null contract $p = (0, 0, 0)$. Clearly, the utility achieved in $p = (0, 0, 0)$ is not $9/80$-optimal, for each possible $\epsilon \in (0, 1/80)$.

[1] Such a learning goal can be seen as instantiating the PAC-learning framework into principal-agent settings.

- $p_{\omega_2} \leq 1/8\epsilon$ and $p \in \mathcal{P}_{a_2}$. Before deriving an upper bound on the principal's expected utility, we observe the following. For any $p \in \mathcal{P}$ with $a^\star(p) = a_2$, we define $\widetilde{p} \in \mathcal{P}$ by $\widetilde{p}_{\omega_1} = 0$ and $\widetilde{p}_\omega = p_\omega$ for all $\omega \neq \omega_1$. Since $F_{a_2,\omega_1} = 0$ and $r_{\omega_1} = 0$, we have $u^{\mathsf{P}}(\widetilde{p}) \geq u^{\mathsf{P}}(p)$ and $a^\star(\widetilde{p}) = a_2$. Hence any utility-maximizing $p' \in \mathcal{P}$ with $a^\star(p') = a_2$ must satisfy $p'_{\omega_1} = 0$. We thus restrict to $p \in \mathcal{P}_{a_2}$ with $p_{\omega_1} = 0$ and $p_{\omega_2} \leq 1/8\epsilon$ that maximize the principal's utility. These constraints can be written as:

$$\begin{cases} \epsilon p_{\omega_2} + (1/2 - \epsilon) p_{\omega_3} - 1/4 \geq 0 \ \wedge \ p_{\omega_2}, p_{\omega_3} \geq 0 \\ p_{\omega_2} \leq 1/8\epsilon. \end{cases}$$

  We notice that the principal's utility is linear over the convex region defined by the latter inequalities. Thus, the maximum is attained in one of the vertices of this region. The vertices of this regions are $p^1 = (0, 0, 1/(2-4\epsilon))$ and $p^2 = (0, 1/8\epsilon, 1/(4-8\epsilon))$. The principal's expected utility evaluated in these contracts is at most:

$$u(p^1) = \sum_{\omega \in \Omega} F_{a_2,\omega}(r_\omega - p^1_\omega) = (1-\epsilon)\left(1 - \frac{1}{2-4\epsilon}\right) < \frac{1}{2}$$

$$u(p^2) = \sum_{\omega \in \Omega} F_{a_2,\omega}(r_\omega - p^2_\omega) \leq -\frac{1}{8} + (1-\epsilon)\left(1 - \frac{1}{4-8\epsilon}\right) < -\frac{1}{8} + \frac{3}{4} = \frac{5}{8}.$$

  Both p^1, p^2 are not $9/80$-optimal, as $\mathrm{OPT} = 3/4 - \epsilon$ with $\epsilon \in (0, 1/80)$.

Consequently, for each action that the agent may take, if $p_{\omega_2} \leq 1/8\epsilon$, then the principal's expected utility cannot be $9/80$-optimal.

We consider two instances with $\epsilon_1 = 1/80N\log(2N)$ and $\epsilon_2 = 1/80N^2$ for some fixed $N > 1$. Let $\mathbb{P}_{\epsilon_1}$ and $\mathbb{P}_{\epsilon_2}$ denote the probability measures induced by the N-round execution of an arbitrary algorithm in the first and second instances, respectively. Then, by applying the Kullback-Leibler decomposition and a simple calculation, we obtain:

$$\begin{aligned}
&\mathcal{KL}(\mathbb{P}_{\epsilon_1}, \mathbb{P}_{\epsilon_2}) \leq \mathbb{E}_{\epsilon_1}\left[\sum_{t=1}^{N} \mathcal{KL}(F^{\epsilon_1}_{a_2}, F^{\epsilon_2}_{a_2})\right] \\
&\leq N\left(\epsilon_1 \log\left(\frac{\epsilon_1}{\epsilon_2}\right) + (1-\epsilon_1)\log\left(\frac{(1-\epsilon_1)}{(1-\epsilon_2)}\right)\right) \\
&\leq N\left(\frac{1}{80N}\underbrace{\frac{\log\left(\frac{N}{\log(2N)}\right)}{\log(2N)}}_{\leq 1} + \left(1 - \frac{1}{80N\log(2N)}\right)\log\left(\frac{1 - \frac{1}{80N\log(2N)}}{1 - \frac{1}{80N^2}}\right)\right) \\
&\leq \frac{1}{80} + N\left(1 - \frac{1}{80N\log(2N)}\right)\underbrace{\log\left(\left(\frac{80N\log(2N)-1}{80N\log(2N)}\right)\left(\frac{80N^2}{80N^2-1}\right)\right)}_{\leq 0} \leq \frac{1}{80}.
\end{aligned}$$

We now introduce the event $\mathcal{I}$, defined as the event in which the final contract returned by a given algorithm satisfies the condition $p_{\omega_2} \geq 1/8\epsilon_2$. We observe that if the event $\mathcal{I}$ holds in the first instance, then the learned contract provides a negative principal's utility. On the contrary, if such an event does not hold in the second instance, the final contract is not $9/80$-optimal, as previously observed. Then, by Bretagnolle-Huber inequality we have that:

$$\mathbb{P}_{\epsilon_2}(\mathcal{I}^c) + \mathbb{P}_{\epsilon_1}(\mathcal{I}) \geq \frac{1}{2}\exp\left(-\mathcal{KL}(\mathbb{P}_{\epsilon_1}, \mathbb{P}_{\epsilon_2})\right) \geq \frac{1}{2}\exp\left(-1/80\right).$$

Consequently, there exists no algorithm returning a $9/80$-optimal with a probability greater or equal to $1 - 1/4\exp\left(-1/80\right)$, thus concluding the proof. □

To overcome the negative result of Theorem 6.1, we focus on designing algorithms for the problem introduced in Definition 6.1. Such a problem relaxes the one introduced above by asking to find a contract whose principal's expected utility is "approximately" close to that of a utility-maximizing contract among those with *bounded* payments.

Definition 6.1 (Learning an optimal bounded contract)**.** *The problem of* learning an optimal bounded contract *reads as follows: Given a bound $B \geq 1$ on payments, an additive approximation error $\rho > 0$, and a probability $\delta \in (0, 1)$, find a contract $p \in [0, B]^m$ such that the following holds:*

$$\mathbb{P}\left(u(p) \geq \max_{p' \in [0,B]^m} u(p') - \rho\right) \geq 1 - \delta.$$

Let us remark that the problem introduced in Definition 6.1 does *not* substantially hinder the generality of the problem of learning an optimal contract. Indeed, since a contract achieving principal's expected utility OPT can be found by means of linear programming [Dutting et al., 2021], it is always the case that an optimal contract uses payments which can be bounded above in terms of the number of bits required to represent the probability distributions over outcomes. Moreover, all the previous works that study the sample complexity of learning optimal contracts focus on settings with bounded payments; see, *e.g.*, [Zhu et al., 2023]. The latter only considers contracts in $[0, 1]^m$, while we address the more general case in which the payments are bounded above by a given $B \geq 1$.

6.2 The `Discover-and-Cover` algorithm

In this section, we provide an algorithm for the problem introduced in Definition 6.1, which we call `Discover-and-Cover` algorithm (Algorithm 6.1). Our main result (Theorem 6.2) is an upper bound on the number of rounds required by the algorithm to return an approximately optimal solution, which grows polynomially in the size of the problem instance when the number of agent's actions n is constant.

The core idea behind Algorithm 6.1 is to learn an optimal bounded contract $p \in [0, B]^m$ by "approximately" identifying the best-response regions $\mathcal{P}_a$. Notice that, since at the end of each round the principal only observes the outcome realized

by the agent's action, rather than the action itself, identifying such best-response regions exactly is *not* possible. Moreover, the principal does *not* even know the set of actions available to the agent, which makes the task even more challenging.

Algorithm 6.1 `Discover-and-Cover`

Require: $\rho \in (0,1)$, $\delta \in (0,1)$, $B \geq 1$

1: Set $\epsilon \leftarrow \frac{\rho^2}{32^2 B m^2 n^2}$
2: Set $\eta \leftarrow \frac{\epsilon\sqrt{m}n}{2}$
3: Set $\alpha \leftarrow \frac{\delta}{2n^3\left[\log\left(\frac{2Bm}{\eta}\right)+\binom{m+n+1}{m}\right]}$
4: Set $q \leftarrow \left\lceil \frac{1}{2\epsilon^2}\log\left(\frac{2m}{\alpha}\right)\right\rceil$
5: $\mathcal{P} \leftarrow \{p \in \mathbb{R}^m_+ \mid \|p\|_1 \leq mB\}$
6: $\mathcal{D} \leftarrow \varnothing$, $\mathcal{F} \leftarrow \varnothing$
7: $\tau \leftarrow 1$
8: **do**
9: $\quad \{\mathcal{L}_d\}_{d\in\mathcal{D}} \leftarrow$ `Try-Cover`()
10: $\quad \tau \leftarrow \tau + 1$
11: **while** $\{\mathcal{L}_d\}_{d\in\mathcal{D}} = \varnothing \wedge \tau \leq 2n$
12: **return** `Find-Contract`$(\{\mathcal{L}_d\}_{d\in\mathcal{D}})$

Algorithm 6.1 builds a set $\mathcal{D}$ of *meta-actions*, where each meta-action $d \in \mathcal{D}$ identifies a set $A(d) \subseteq \mathcal{A}$ of one or more (unknown) agent's actions. A meta-action groups together "similar" actions, in the sense that they induce similar distributions over outcomes. The algorithm incrementally discovers new elements of $\mathcal{D}$ by trying to cover a suitably-defined set $\mathcal{P}$ of contracts (see Line 5) by means of *approximate best-response regions* $\mathcal{L}_d$, one for each $d \in \mathcal{D}$. In particular, $\mathcal{L}_d$ is a suitably-defined polytope that "approximately describes" the union of all the best-response regions $\mathcal{P}_a$ of actions $a \in A(d)$ associated with d. Each time $\mathcal{D}$ is updated, the algorithm calls the `Try-Cover` procedure (Algorithm 6.3), whose aim is to cover $\mathcal{P}$ with approximate best-response regions. `Try-Cover` works by iteratively finding hyperplanes that separate such regions, by "testing" suitable contracts through the `Action-Oracle` procedure (Algorithm 6.2). Given a contract $p \in \mathcal{P}$, `Action-Oracle` checks whether it is possible to safely map the agent's best response $a^\star(p)$ to an already-discovered meta-action $d \in \mathcal{D}$ or *not*. In the latter case, `Action-Oracle` refines the set $\mathcal{D}$ by either adding a new meta-action or merging a group of meta-actions if their associated actions can be considered as "similar". Whenever the set $\mathcal{D}$ changes, `Try-Cover` is terminated (returning $\varnothing$). The algorithm continues trying to cover $\mathcal{P}$ until a satisfactory cover is found. Finally, such a cover is used to compute a contract $p \in [0, B]^m$ to be returned, by means of the procedure `Find-Contract` (Algorithm 6.5).

In the rest of this section, we describe the details of all the procedures used by Algorithm 6.1. In particular, Section 6.2.1 is concerned with `Action-Oracle`, Section 6.2.2 with `Try-Cover`, and Section 6.2.4 with `Find-Contract`. Finally, Section 6.2.5 concludes the analysis of Algorithm 6.1 by putting everything together so as to prove the guarantees of the algorithm.

6.2.1 The `Action-Oracle` procedure

In this section, we present the `Action-Oracle` procedure. This procedure plays a crucial role when Algorithm 6.1 attempts to identify a cover of the space of contracts in approximate best-response regions by means of Algorithm 6.3. Intuitively, this is because the `Action-Oracle` procedure assigns a meta-action to each contract, allowing Algorithm 6.3 to partition the space of contracts into approximate best-response regions. Specifically, given a contract $p \in \mathcal{P}$ as input, the `Action-Oracle` procedure (Algorithm 6.2) determines whether it is possible to safely map the agent's best response $a^\star(p)$ to some meta-action $d \in \mathcal{D}$. If this is the case, then the procedure returns such a meta-action. Otherwise, the procedure has to properly refine the set $\mathcal{D}$ of meta-actions, as we describe in the rest of this subsection.

In the first phase (Lines 1–6), Algorithm 6.2 prescribes the principal to commit to the contract $p \in \mathcal{P}$ given as input for $q \in \mathbb{N}$ consecutive rounds. This is done to build an empirical distribution $\widetilde{F} \in \Delta_\Omega$ that estimates the (true) distribution over outcomes $F_{a^\star(p)}$ induced by the agent's best response $a^\star(p)$. In the second phase (Lines 7–20), such an empirical distribution is compared with those previously computed, in order to decide whether $a^\star(p)$ can be safely mapped to some $d \in \mathcal{D}$ or *not*. To perform such a comparison, all the empirical distributions computed by the algorithm are stored in a dictionary $\mathcal{F}$ (initialized in Line 6 of Algorithm 6.1), where $\mathcal{F}[d]$ contains all the $\widetilde{F} \in \Delta_\Omega$ associated with $d \in \mathcal{D}$.

The first phase of Algorithm 6.2 ensures that $\|\widetilde{F} - F_{a^\star(p)}\|_\infty \leq \epsilon$ holds with sufficiently high probability. For ease of presentation, we introduce a *clean event* that allows us to factor out from our analysis the "with high probability" statements.

Definition 6.2 (Clean event)**.** *Let $\epsilon > 0$ be a constant. We define $\mathcal{E}_\epsilon$ as the event under which, whenever Algorithm 6.2 is invoked during the execution of Algorithm 6.1-receiving as input any possible $p \in \mathcal{P}$ it computes some $\widetilde{F} \in \Delta_\Omega$ that satisfies $\|\widetilde{F} - F_{a^\star(p)}\|_\infty \leq \epsilon$.*

In the second phase, Algorithm 6.2 searches for all the $d \in \mathcal{D}$ such that $\mathcal{F}[d]$ contains *at least one* empirical distribution which is "sufficiently close" to the $\widetilde{F}$ that has just been computed by the algorithm. Formally, the algorithm looks for all the $d \in \mathcal{D}$ such that $\|F - \widetilde{F}\|_\infty \leq 2\epsilon$ for at least one $F \in \mathcal{F}[d]$. Then, three cases are possible:

(i) If the algorithm finds a *unique* $d \in \mathcal{D}$ with such a property (case $|\mathcal{D}^\diamond| = 1$), then d is returned since $a^\star(p)$ can be safely mapped to d.

(ii) If the algorithm does *not* find any $d \in \mathcal{D}$ with such a property (case $\mathcal{D}^\diamond = \varnothing$), then a new meta-action $d^\diamond$ is added to $\mathcal{D}$.

(iii) If the algorithm finds *more than one* $d \in \mathcal{D}$ with such a property ($|\mathcal{D}^\diamond| > 1$), then all the meta-actions in $\mathcal{D}^\diamond$ are merged into a single (new) meta-action $d^\diamond$.

In Algorithm 6.2, the last two cases above are jointly managed by Lines 17–20, and in both cases the algorithm returns the special value $\perp$. This alerts the calling procedure that the set $\mathcal{D}$ has changed, and, thus, the `Try-Cover` procedure needs to be re-started. Notice that Algorithm 6.2 also takes care of properly updating the

Algorithm 6.2 Action-Oracle

Require: $p \in \mathcal{P}$
1: $I_\omega \leftarrow 0$ for all $\omega \in \Omega$ ▷ Phase 1: Estimate
2: **for** $\tau = 1, \dots, q$ **do**
3: Commit to p and observe $\omega \in \Omega$
4: $I_\omega \leftarrow I_\omega + 1$
5: **end for**
6: Build $\widetilde{F} \in \Delta_\Omega : \widetilde{F}_\omega = I_\omega / q \; \forall \omega \in \Omega$
7: $\mathcal{D}^\diamond \leftarrow \varnothing$ ▷ Phase 2: Compare
8: **for** $d \in \mathcal{D}$ **do**
9: **if** $\exists F \in \mathcal{F}[d] : \|F - \widetilde{F}\|_\infty \leq 2\epsilon$ **then**
10: $\mathcal{D}^\diamond \leftarrow \mathcal{D}^\diamond \cup \{d\}$
11: **end if**
12: **end for**
13: **if** $|\mathcal{D}^\diamond| = 1$ **then**
14: $\mathcal{F}[d] \leftarrow \mathcal{F}[d] \cup \{\widetilde{F}\}$ ▷ unique $d \in \mathcal{D}^\diamond$
15: **return** d
16: **end if**
17: $\mathcal{D} \leftarrow (\mathcal{D} \setminus \mathcal{D}^\diamond) \cup \{d^\diamond\}$ ▷ $d^\diamond$ is new
18: $\mathcal{F}[d^\diamond] \leftarrow \bigcup_{d \in \mathcal{D}^\diamond} \mathcal{F}[d] \cup \{\widetilde{F}\}$ ▷ $\widetilde{F}_{d^\diamond} = \widetilde{F}$ denotes a representative distribution of $d^\diamond$
19: Clear $\mathcal{F}[d]$ for all $d \in \mathcal{D}^\diamond$
20: **return** $\perp$

dictionary $\mathcal{F}$, which is done in Lines 14 and 19. Moreover, whenever Algorithm 6.2 adds a new meta-action $d^\diamond$ into $\mathcal{D}$, this is also associated with a particular empirical distribution $\widetilde{F}_{d^\diamond}$, which is set to be equal to $\widetilde{F}$ (Line 18). We remark that the symbols $\widetilde{F}_d$ for $d \in \mathcal{D}$ have only been introduced for ease of exposition, and they do *not* reference actual variables declared in the algorithm. Operationally, one can replace each appearance of $\widetilde{F}_d$ in the algorithms with any fixed empirical distribution contained in the dictionary entry $\mathcal{F}[d]$.

The first crucial property that Algorithm 6.2 guarantees is that *only "similar" agent's best-response actions are mapped to the same meta-action*. In order to formally state such a property, we first need to introduce the definition of set of agent's actions *associated with* a meta-action in $\mathcal{D}$.

Definition 6.3 (Associated actions)**.** *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, we let $A : \mathcal{D} \to 2^{\mathcal{A}}$ be a function such that $A(d)$ represents the* set of actions associated with *the meta-action $d \in \mathcal{D}$, defined as follows:*

$$A(d) = \left\{ a \in \mathcal{A} \mid \|\widetilde{F}_d - F_a\|_\infty \leq 5\epsilon n \wedge \exists p \in \mathcal{P} : a = a^\star(p) \right\}.$$

The set $A(d)$ encompasses all the agent's actions $a \in \mathcal{A}$ whose distributions over outcomes F_a are "similar" to the empirical distribution $\widetilde{F}_d$ associated with the meta-action $d \in \mathcal{D}$, where the entity of the similarity is defined in a suitable way depending on ϵ and the number of agent's actions n. Moreover, notice that the set $A(d)$ only contains agent's actions that can be induced as best response for at least one contract $p \in \mathcal{P}$. This is needed in order to simplify the analysis of the

algorithm. Let us also remark that an agent's action may be associated with *more than one* meta-action. Equipped with Definition 6.3, the property introduced above can be formally stated as follows:

Lemma 6.1. *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, under the event $\mathcal{E}_\epsilon$, if Algorithm 6.2 returns a meta-action $d \in \mathcal{D}$ for a contract $p \in \mathcal{P}$ given as input, then it holds that $a^\star(p) \in A(d)$.*

Proof. For ease of presentation, in this proof we adopt the following additional notation. Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, for every meta-action $d \in \mathcal{D}$, we let $\mathcal{P}_d \subseteq \mathcal{P}$ be the set of all the contracts that have been provided as input to Algorithm 6.2 during an execution in which it computed an empirical distribution that belongs to $\mathcal{F}[d]$. Moreover, we let $I(d) \subseteq \mathcal{A}$ be the set of all the agent's actions that have been played as a best response by the agent during at least one of such executions of Algorithm 6.2. Formally, by exploiting the definition of $\mathcal{P}_d$, we can write $I(d) := \bigcup_{p \in \mathcal{P}_d} a^\star(p)$.

First, we prove the following crucial property of Algorithm 6.2:

Property 1. *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, under the event $\mathcal{E}_\epsilon$, it holds that $I(d) \cap I(d') = \varnothing$ for every pair of (different) meta-actions $d \neq d' \in \mathcal{D}$.*

In order to show that Property 1 holds, we assume by contradiction that there exist two different meta-actions $d \neq d' \in \mathcal{D}$ such that $a \in I(d)$ and $a \in I(d')$ for some agent's action $a \in \mathcal{A}$, namely that $I(d) \cap I(d') \neq \varnothing$. This implies that there exist two contracts $p \in \mathcal{P}_d$ and $p' \in \mathcal{P}_{d'}$ such that $a^\star(p) = a^\star(p') = a$. Let $\widetilde{F} \in \mathcal{F}[d]$ and $\widetilde{F}' \in \mathcal{F}[d']$ be the empirical distributions over outcomes computed by Algorithm 6.2 when given as inputs the contracts p and p', respectively. Then, under the event $\mathcal{E}_\epsilon$, we have that $\|\widetilde{F} - \widetilde{F}'\|_\infty \leq 2\epsilon$, since the two empirical distributions are generated by sampling from the same (true) distribution F_a. Clearly, the way in which Algorithm 6.2 works implies that such empirical distributions are associated with the same meta-action and put in the same entry of the dictionary $\mathcal{F}$. This contradicts the assumption that $d \neq d'$, proving that Property 1 holds.

Next, we show the following additional crucial property of Algorithm 6.2:

Property 2. *Let $\mathcal{D}$ be a set of meta-actions and $\mathcal{F}$ be a dictionary of empirical distributions computed by means of Algorithm 6.2. Suppose that the following holds for every $d \in \mathcal{D}$ and $a, a' \in I(d)$:*

$$\|F_a - F_{a'}\|_\infty \leq 4\epsilon(|I(d)| - 1). \tag{6.1}$$

Then, if Algorithm 6.2 is run again, under the event $\mathcal{E}_\epsilon$ the same condition continues to hold for the set of meta-actions and the dictionary obtained after the execution of Algorithm 6.2.

For ease of presentation, in the following we call $\mathcal{D}^{\text{old}}$ and $\mathcal{F}^{\text{old}}$ the set of meta-actions and the dictionary of empirical distributions, respectively, before the last execution of Algorithm 6.2, while we call $\mathcal{D}^{\text{new}}$ and $\mathcal{F}^{\text{new}}$ those obtained after the last run of Algorithm 6.2. Moreover, in order to avoid any confusion, given $\mathcal{D}^{\text{old}}$ and $\mathcal{F}^{\text{old}}$, we write $I^{\text{old}}(d)$ in place of $I(d)$ for any meta-action $d \in \mathcal{D}^{\text{old}}$. Similarly, given $\mathcal{D}^{\text{new}}$ and $\mathcal{F}^{\text{new}}$, we write $I^{\text{new}}(d)$ in place of $I(d)$ for any $d \in \mathcal{D}^{\text{new}}$.

In order to show that Property 2 holds, let $p \in \mathcal{P}$ be the contract given as input to Algorithm 6.2 during its last run. Next, we prove that, no matter how Algorithm 6.2 updates $\mathcal{D}^{\text{old}}$ and $\mathcal{F}^{\text{old}}$ in order to obtain $\mathcal{D}^{\text{new}}$ and $\mathcal{F}^{\text{new}}$, the condition in Property 2 continues to hold. We split the proof in three cases.

1. If $|\mathcal{D}^{\diamond}| = 0$, then $\mathcal{D}^{\text{new}} = \mathcal{D}^{\text{old}} \cup \{d^{\diamond}\}$, where $d^{\diamond}$ is a new meta-action. Since $\mathcal{F}^{\text{new}}$ is built by adding a new entry $\mathcal{F}^{\text{new}}[d^{\diamond}] = \{\widetilde{F}\}$ to $\mathcal{F}^{\text{old}}$ while leaving all the other entries unchanged, it holds that $I^{\text{new}}(d^{\diamond}) = \{a^{\star}(p)\}$ and $I^{\text{new}}(d) = I^{\text{old}}(d)$ for all $d \in \mathcal{D}^{\text{old}}$. As a result, the condition in Equation (6.1) continues to hold for all the meta-actions $d \in \mathcal{D}^{\text{new}} \setminus \{d^{\diamond}\}$. Moreover, for the meta-action $d^{\diamond}$, the following holds:

$$0 = \|F_{a^{\star}(p)} - F_{a^{\star}(p)}\|_{\infty} \leq 4\epsilon(|I^{\text{new}}(d^{\diamond})| - 1) = 0,$$

   as $|I^{\text{new}}(d^{\diamond})| = 1$. This proves that the condition in Equation (6.1) also holds for $d^{\diamond}$.

2. If $|\mathcal{D}^{\diamond}| = 1$, then $\mathcal{D}^{\text{new}} = \mathcal{D}^{\text{old}}$. We distinguish between two cases.

   (a) In the case in which $a^{\star}(p) \in \bigcup_{d \in \mathcal{D}^{\text{old}}} I^{\text{old}}(d)$, Property 1 immediately implies that $I^{\text{new}}(d) = I^{\text{old}}(d)$ for all $d \in \mathcal{D}^{\text{new}} = \mathcal{D}^{\text{old}}$. Indeed, if this is *not* the case, then there would be two different meta-actions $d \neq d' \in \mathcal{D}^{\text{new}} = \mathcal{D}^{\text{old}}$ such that $I^{\text{new}}(d) = I^{\text{old}}(d) \cup \{a^{\star}(p)\}$ (since $\mathcal{F}^{\text{new}}[d] = \mathcal{F}^{\text{old}}[d] \cup \{\widetilde{F}\}$) and $a^{\star}(p) \in I^{\text{new}}(d') = I^{\text{old}}(d')$, contradicting Property 1. As a result, the condition in Equation (6.1) continues to hold for all the meta-actions after the execution of Algorithm 6.2.

   (b) In the case in which $a^{\star}(p) \notin \bigcup_{d \in \mathcal{D}^{\text{old}}} I^{\text{old}}(d)$, the proof is more involved. Let $d \in \mathcal{D}^{\text{new}} = \mathcal{D}^{\text{old}}$ be the (unique) meta-action in the set $\mathcal{D}^{\diamond}$ computed by Algorithm 6.2. Notice that $I^{\text{new}}(d) = I^{\text{old}}(d) \cup \{a^{\star}(p)\}$ by how Algorithm 6.2 works. As a first step, we show that, for every pair of actions $a, a' \in I^{\text{new}}(d)$ with $a = a^{\star}(p)$, Equation (6.1) holds. Under the event $\mathcal{E}_{\epsilon}$, it holds that $\|F_{a^{\star}(p)} - \widetilde{F}\|_{\infty} \leq \epsilon$, where $\widetilde{F}$ is the empirical distribution computed by Algorithm 6.2. Moreover, since the meta-action d has been added to $\mathcal{D}^{\diamond}$ by Algorithm 6.2, there exists an empirical distribution $F \in \mathcal{F}[d]$ such that $\|\widetilde{F} - F\|_{\infty} \leq 2\epsilon$, and, under the event $\mathcal{E}_{\epsilon}$, there exists an action $a'' \in I^{\text{old}}(d)$ such that $\|F - F_{a''}\|_{\infty} \leq \epsilon$. Then, by applying the triangular inequality we can

show that:

$$\|F_{a^\star(p)} - F_{a''}\|_\infty \leq \|F_{a^\star(p)} - \widetilde{F}\|_\infty + \|\widetilde{F} - F\|_\infty + \|F - F_{a''}\|_\infty \leq 4\epsilon.$$

By using the fact that the condition in Equation (6.1) holds for $\mathcal{D}^{\text{new}}$ and $\mathcal{F}^{\text{new}}$, for every action $a' \in I^{\text{new}}(d)$ it holds that:

$$\begin{aligned}\|F_{a^\star(p)} - F_{a'}\|_\infty &\leq \|F_{a^\star(p)} - F_{a''}\|_\infty + \|F_{a''} - F_{a'}\|_\infty \\ &\leq 4\epsilon(|I^{\text{old}}(d)| - 1) + 4\epsilon \\ &\leq 4\epsilon|I^{\text{old}}(d)| \\ &= 4\epsilon(|I^{\text{new}}(d)| - 1),\end{aligned}$$

where the last equality holds since $|I^{\text{new}}(d)| = |I^{\text{old}}(d)| + 1$, as $a^\star(p) \notin \bigcup_{d \in \mathcal{D}^{\text{old}}} I^{\text{old}}(d)$. This proves that Equation (6.1) holds for every pair of actions $a, a' \in I^{\text{new}}(d)$ with $a = a^\star(p)$. For all the other pairs of actions $a, a' \in I^{\text{new}}(d)$, the equation holds since it was already satisfied before the last execution of Algorithm 6.2. Analogously, given that $I^{\text{new}}(d) = I^{\text{old}}(d)$ for all the meta-actions in $\mathcal{D}^{\text{new}} \setminus \{d\}$, we can conclude that the condition in Equation (6.1) continues to holds for such meta-actions as well.

3. If $|\mathcal{D}^\diamond| > 1$, then $\mathcal{D}^{\text{new}} = \left(\mathcal{D}^{\text{old}} \setminus \mathcal{D}^\diamond\right) \cup \{d^\diamond\}$, where $d^\diamond$ is a new meta-action. Clearly, the condition in Equation (6.1) continues to hold for all the meta-actions in $\mathcal{D}^{\text{old}} \setminus \mathcal{D}^\diamond$. We only need to show that the condition holds for $d^\diamond$. We distinguish between two cases.

   (a) In the case in which $a^\star(p) \in \bigcup_{d \in \mathcal{D}^{\text{old}}} I^{\text{old}}(d)$, let us first notice that $a^\star(p) \in I^{\text{old}}(d)$ for some $d \in \mathcal{D}^\diamond$, otherwise Property 1 would be violated (as $a^\star(p) \in I^{\text{new}}(d^\diamond)$ by definition). Moreover, it is easy to see that $I^{\text{new}}(d^\diamond) = \bigcup_{d \in \mathcal{D}^\diamond} I^{\text{old}}(d)$ and, additionally, $I^{\text{old}}(d) \cap I^{\text{old}}(d') = \varnothing$ for all $d \neq d' \in \mathcal{D}^\diamond$, given how Algorithm 6.2 works and thanks to Property 1. In the following, for ease of presentation, we assume w.l.o.g. that the meta-actions in $\mathcal{D}^\diamond$ are indexed by natural numbers so that $\mathcal{D}^\diamond := \{d_1, \dots d_{|\mathcal{D}^\diamond|}\}$ and $a^\star(p) \in I^{\text{old}}(d_1)$. Next, we show that Equation (6.1) holds for every pair $a, a' \in I^{\text{new}}(d^\diamond)$. First, by employing an argument similar to the one used to prove Point 2.2, we can show that, for every $d_j \in \mathcal{D}^\diamond$ with $j > 1$, there exists an action $a'' \in I^{\text{old}}(d_j)$ such that $\|F_{a^\star(p)} - F_{a''}\|_\infty \leq 4\epsilon$. Then, for every pair of actions $a, a' \in I^{\text{new}}(d^\diamond)$ such that $a \in I^{\text{old}}(d_1)$ and $a' \in I^{\text{old}}(d_j)$ for some $d_j \in \mathcal{D}^\diamond$ with $j > 1$, the following holds:

$$\begin{aligned}\|F_a - F_{a'}\|_\infty &\leq \|F_a - F_{a^\star(p)}\|_\infty + \|F_{a^\star(p)} - F_{a''}\|_\infty + \|F_{a''} - F_{a'}\|_\infty \\ &\leq 4\epsilon(|I^{\text{old}}(d_1)| - 1) + 4\epsilon + 4\epsilon(|I^{\text{old}}(d_j)| - 1) \\ &= 4\epsilon(|I^{\text{old}}(d_1)| + |I^{\text{old}}(d_j)| - 1) \\ &\leq 4\epsilon(|I^{\text{new}}(d^\diamond)| - 1),\end{aligned}$$

where the first inequality follows from an application of the triangular inequality, the second one from the fact that Equation (6.1) holds before the last execution of Algorithm 6.2, while the last inequality holds since $|I^{\text{new}}(d^\diamond)| = \sum_{d \in \mathcal{D}^\diamond} |I^{\text{old}}(d)|$ given that $I^{\text{old}}(d) \cap I^{\text{old}}(d') = \varnothing$ for all $d \neq d' \in \mathcal{D}^\diamond$ thanks to Property 1. As a final step, we show that Equation (6.1) holds for every pair of actions $a, a' \in I^{\text{new}}(d^\diamond)$ such that $a \in I^{\text{old}}(d_i)$ and $a' \in I^{\text{old}}(d_j)$ for some $d_i, d_j \in \mathcal{D}^\diamond$ with $i \neq j > 1$. By the fact that $d_i, d_j \in \mathcal{D}^\diamond$ and triangular inequality, it follows that there exist $a'' \in I^{\text{old}}(d_i)$ and $a''' \in I^{\text{old}}(d_j)$ such that $\|F_{a''} - \widetilde{F}\|_\infty \leq 3\epsilon$ and $\|F_{a'''} - \widetilde{F}\|_\infty \leq 3\epsilon$ under $\mathcal{E}_\epsilon$. Then, with steps similar to those undertaken above, we can prove the following:

$$\begin{aligned}\|F_a - F_{a'}\|_\infty &\leq \|F_a - F_{a''}\|_\infty + \|F_{a''} - \widetilde{F}\|_\infty + \|\widetilde{F} - F_{a'''}\|_\infty + \|F_{a'''} - F_{a'}\|_\infty \\ &\leq 4\epsilon(|I^{\text{old}}(d_i)| - 1) + 3\epsilon + 3\epsilon + 4\epsilon(|I^{\text{old}}(d_j)| - 1) \\ &= 4\epsilon(|I^{\text{old}}(d_i)| + |I^{\text{old}}(d_j)| - 1) + 2\epsilon \\ &\leq 4\epsilon(|I^{\text{new}}(d^\diamond)| - 1),\end{aligned}$$

which shows that Equation (6.1) is satisfied for all the pairs of actions $a, a' \in I^{\text{new}}(d^\diamond)$.

(b) In the case in which $a^\star(p) \notin \bigcup_{d \in \mathcal{D}^{\text{old}}} I^{\text{old}}(d)$, let us first observe that $I^{\text{new}}(d^\diamond) = \bigcup_{d \in \mathcal{D}^\diamond} I^{\text{old}}(d) \cup \{a^\star(p)\}$ and $I^{\text{old}}(d) \cap I^{\text{old}}(d') = \varnothing$ for all $d \neq d' \in \mathcal{D}^\diamond$, given how Algorithm 6.2 works and thanks to Property 1. Next, we show that Equation (6.1) holds for every pair of actions $a, a' \in I^{\text{new}}(d^\diamond)$. As a first step, we consider the case in which $a = a^\star(p)$, and we show that Equation (6.1) holds for every $a' \in I^{\text{new}}(d^\diamond)$ such that $a' \in I^{\text{old}}(d)$ for some $d \in \mathcal{D}^\diamond$. In order to show this, we first observe that, given how Algorithm 6.2 works, there exists $F \in \mathcal{F}^{\text{old}}[d]$ such that $\|\widetilde{F} - F\| \leq 2\epsilon$, and, under the event $\mathcal{E}_\epsilon$, there exists an action $a'' \in I^{\text{old}}(d)$ such that $\|F_{a''} - F\| \leq \epsilon$. Then:

$$\begin{aligned}\|F_{a^\star(p)} - F_{a'}\|_\infty &\leq \|F_{a^\star(p)} - \widetilde{F}\|_\infty + \|\widetilde{F} - F\|_\infty + \|F - F_{a''}\|_\infty + \|F_{a''} - F_{a'}\|_\infty \\ &\leq 4\epsilon + 4\epsilon(|I^{\text{old}}(d)| - 1) \\ &\leq 4\epsilon(|I^{\text{new}}(d^\diamond)| - 1),\end{aligned}$$

where the last inequality holds since $|I^{\text{new}}_{d^\diamond}| = \sum_{d \in \mathcal{D}^\diamond} |I^{\text{old}}(d)|$ as $I^{\text{old}}(d) \cap I^{\text{old}}(d') = \varnothing$ for all $d \neq d' \in \mathcal{D}^\diamond$. As a second step, we consider the case in which $a \in I^{\text{old}}(d)$ and $a' \in I^{\text{old}}(d')$ for some pair $d \neq d' \in \mathcal{D}^\diamond$. Given how Algorithm 6.2 works, there exist $F \in \mathcal{F}^{\text{old}}[d]$ and $F' \in \mathcal{F}^{\text{old}}[d']$ such that $\|F - \widetilde{F}\| \leq 2\epsilon$ and $\|F' - \widetilde{F}\| \leq 2\epsilon$. Moreover, by the fact that $d, d' \in \mathcal{D}^\diamond$ and the triangular inequality, it follows that there exist $a'' \in I^{\text{old}}(d)$ and $a''' \in I^{\text{old}}(d')$ such that $\|F_{a''} - \widetilde{F}\|_\infty \leq 3\epsilon$ and

$\|F_{a'''} - \widetilde{F}\|_\infty \leq 3\epsilon$ under the event $\mathcal{E}_\epsilon$. Then, the following holds:

$$\begin{aligned}\|F_a - F_{a'}\|_\infty &\leq \|F_a - F_{a''}\|_\infty + \|F_{a''} - \widetilde{F}\|_\infty + \|\widetilde{F} - F_{a'''}\|_\infty + \|F_{a'''} - F_{a'}\|_\infty \\ &\leq 4\epsilon(|I^{\text{old}}(d)| - 1) + 3\epsilon + 3\epsilon + 4\epsilon(|I^{\text{old}}(d')| - 1) \\ &= 4\epsilon(|I^{\text{old}}(d)| + |I^{\text{old}}(d')| - 1) + 2\epsilon \\ &\leq 4\epsilon(|I^{\text{new}}(d^\diamond)| - 1).\end{aligned}$$

Finally, by observing that the size of $I(d)$ for each possible set of meta-actions $\mathcal{D}$ is bounded by the number of agent's actions n, for each $a, a' \in I(d)$ and $d \in \mathcal{D}$ we have that:

$$\|F_a - F_{a'}\| \leq 4n\epsilon. \tag{6.2}$$

Then, under the event $\mathcal{E}_\epsilon$, by letting $a' \in I(d)$ be the action leading to the empirical distribution $\widetilde{F}_d$ computed at Line 18 in Algorithm 6.2 we have $\|\widetilde{F}_d - F_{a'}\|_\infty \leq \epsilon$. Finally, combining the two inequalities, we get:

$$\|\widetilde{F}_d - F_{a^\star(p)}\|_\infty \leq \|\widetilde{F}_d - F_{a'}\|_\infty + \|F_{a'} - F_{a^\star(p)}\|_\infty \leq 4\epsilon n + \epsilon \leq 5\epsilon n.$$

This implies that $a^\star(p) \in A(d)$.

To conclude the proof we show that $I(d) \subseteq A(d)$. Let $a' \in I(d)$ be an arbitrary action, and let $\widetilde{F}_d$ be the empirical distribution computed by Algorithm 6.2 by sampling from F_a. Then for each $a'' \in I(d)$ we have:

$$\|\widetilde{F}_d - F_{a''}\|_\infty \leq \|\widetilde{F}_d - F_{a'}\|_\infty + \|F_{a'} - F_{a''}\|_\infty \leq 4\epsilon n + \epsilon \leq 5\epsilon n.$$

Since the latter argument holds for all the actions $a \in I(d)$ this hows that

$$\|F_a - F_{a'}\|_\infty \leq 4n\epsilon, \tag{6.3}$$

for all the actions $a, a' \in A(d)$. □

Lemma 6.1 follows from the observation that, under the event $\mathcal{E}_\epsilon$, the empirical distributions computed by Algorithm 6.2 are "close" to the true ones, and, thus, the distributions over outcomes of all the actions associated with d are sufficiently "close" to each other. A non-trivial part of the proof of Lemma 6.1 is to show that Algorithm 6.2 does *not* put in the same entry $\mathcal{F}[d]$ empirical distributions that form a "chain" growing arbitrarily in terms of $\|\cdot\|_\infty$ norm, but instead the length of such "chains" is always bounded by $5\epsilon n$.

The second crucial property is made formal by the following lemma:

Lemma 6.2. *Under the event $\mathcal{E}_\epsilon$, during the execution of Algorithm 6.1, Algorithm 6.2 returns $\perp$ at most $2n$ times.*

Proof. We first observe that under the event $\mathcal{E}_\epsilon$, the size of $\mathcal{D}$ increases whenever the principal observes an empirical distribution $\widetilde{F}$ such that $\|\widetilde{F} - \widetilde{F}'\|_\infty \geq 2\epsilon$ for every $\widetilde{F}' \in \mathcal{F}$. This condition holds when the agent chooses an action that has not been selected before. This is because, if the principal commits to a contract implementing an action the agent has already played, the resulting estimated

empirical distribution $\widetilde{F}$ must satisfy $\|\widetilde{F}-F\|_\infty \leq 2\epsilon$ for some $d \in \mathcal{D}$ and $F \in \mathcal{F}_d$, as guaranteed by the event $\mathcal{E}_\epsilon$. Consequently, the cardinality of $\mathcal{D}$ can increase at most n times, which corresponds to the total number of actions. Furthermore, we observe that the cardinality of $\mathcal{D}$ can decrease by merging one or more meta-actions into a single one, with the condition $|\mathcal{D}| \geq 1$. Therefore, in the worst case the cardinality of $\mathcal{D}$ is reduced by one n times, resulting in $2n$ being the maximum number of times Algorithm 6.2 returns $\perp$. □

Intuitively, Lemma 6.2 follows from the fact that Algorithm 6.2 increases the cardinality of the set $\mathcal{D}$ by one only when $\|F - \widetilde{F}\|_\infty > 2\epsilon$ for all $F \in \mathcal{F}[d]$ and $d \in \mathcal{D}$. In such a case, under the event $\mathcal{E}_\epsilon$ the agent's best response is an action that has never been played before. Thus, the cardinality of $\mathcal{D}$ can be increased at most n times. Moreover, in the worst case the cardinality of $\mathcal{D}$ is reduced by one for n times, resulting in $2n$ being the maximum number of times Algorithm 6.2 returns $\perp$ during the execution of Algorithm 6.1. In the following, we formally introduce the definition of *cost of a meta-action*.

Definition 6.4 (Cost of a meta-action)**.** *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, we let $c : \mathcal{D} \to [0, 1]$ be a function such that $c(d)$ represents the* cost of meta-action *$d \in \mathcal{D}$, defined as $c(d) = \min_{a \in A(d)} c_a$.*

Then, by Holder's inequality, we can prove that the two following lemmas hold:

Lemma 6.3. *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, under the event $\mathcal{E}_\epsilon$, for every meta-action $d \in \mathcal{D}$ and associated action $a \in A(d)$ it holds that $|c(d) - c_a| \leq 4B\epsilon mn$.*

Proof. Let $d \in \mathcal{D}$ be a meta-action and assume that the event $\mathcal{E}_\epsilon$ holds. Let $a' \in \arg\min_{a \in A(d)} c_a$ and let p' be a contract such that $a^\star(p') = a'$. By Definition 6.3, we know that for any action $a \in A(d)$ there exists a contract $p \in \mathcal{P}$ such that $a^\star(p) = a$. Then, the following inequalities hold:

$$4B\epsilon mn \geq \|F_a - F_{a'}\|_\infty \|p\|_1 \geq \sum_{\omega \in \Omega} (F_{a,\omega} - F_{a',\omega})\, p_\omega \geq c_a - c_{a'}, \tag{6.4}$$

and, similarly,

$$4B\epsilon mn \geq \|F_{a'} - F_a\|_\infty \|p'\|_1 \geq \sum_{\omega \in \Omega} (F_{a',\omega} - F_{a,\omega})\, p'_\omega \geq c_{a'} - c_a. \tag{6.5}$$

In particular, in the chain of inequalities above, the first '$\geq$' holds because the inequality $\|F_a - F_{a'}\|_\infty \leq 4\epsilon n$ follows from Lemma 6.1, and because the $\|\cdot\|_1$ norm of contracts in $\mathcal{P}$ is bounded above by Bm. Finally, by applying Equations (6.4) and (6.5), we have $|c_a - c_{a'}| = |c_a - c(d)| \leq 4B\epsilon mn$, for every possible action $a \in A(d)$ since $c_{a'} = c(d)$ by definition, concluding the proof. □

Lemma 6.4. *Given a set $\mathcal{D}$ of meta-actions and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, under the event $\mathcal{E}_\epsilon$, for every meta-action $d \in \mathcal{D}$, associated action $a \in A(d)$, and contract $p \in \mathcal{P}$ it holds*

$$\left| \sum_{\omega \in \Omega} \widetilde{F}_{d,\omega}\, p_\omega - c(d) - \sum_{\omega \in \Omega} F_{a,\omega}\, p_\omega + c_a \right| \leq 9B\epsilon mn.$$

Proof. To prove the lemma we observe that for each action $a \in A(d)$ it holds:

$$\begin{aligned} \left| \sum_{\omega \in \Omega} \widetilde{F}_{d,\omega} p_\omega - c(d) - \sum_{\omega \in \Omega} F_{a,\omega} p_\omega + c_a \right| &\leq \|\widetilde{F}_d - F_a\|_\infty \|p\|_1 + |c(d) - c_a| \\ &\leq 5B\epsilon mn + 4B\epsilon mn = 9B\epsilon mn. \end{aligned}$$

The first inequality holds by applying the triangular inequality and Holder's inequality. The second inequality holds by employing Lemma 6.3 and leveraging Definition 6.3 that guarantees $\|\widetilde{F}_d - F_a\|_\infty \leq 5\epsilon n$ for every action $a \in A(d)$. Additionally, we observe that the $\|\cdot\|_1$ norm of contracts in $\mathcal{P}$ is upper bounded by Bm, which concludes the proof. □

We conclude by providing an auxiliary lemmas related to the `Action-Oracle` procedure that will be employed to bound the probability of the clean event $\mathcal{E}_\epsilon$ in the proof of Theorem 6.2.

Lemma 6.5. *Given $\lambda, \beta \in (0, 1)$, let $\widetilde{F} \in \Delta_\Omega$ be the empirical distribution computed by Algorithm 6.2 with $q := \lceil 1/2\beta^2 \log(2m/\lambda) \rceil$ for a contract $p \in \mathcal{P}$ given as input. Then, it holds that $\mathbb{E}[\widetilde{F}_\omega] = F_{a^\star(p),\omega}$ for all $\omega \in \Omega$ and, with probability of at least $1 - \lambda$, it also holds that $\|\widetilde{F} - F_{a^\star(p)}\|_\infty \leq \beta$.*

Proof. By construction, the empirical distribution $\widetilde{F} \in \Delta_\Omega$ is computed from q *i.i.d.* samples drawn according to $F_{a^\star(p)}$, with each $\omega \in \Omega$ being drawn with probability $F_{a^\star(p),\omega}$.

Therefore, the empirical distribution $\widetilde{F}$ is a random vector supported on Δ_Ω, whose expectation is such that $\mathbb{E}[\widetilde{F}_\omega] = F_{a^\star(p),\omega}$ for all $\omega \in \Omega$. Moreover, by Hoeffding's inequality, for every $\omega \in \Omega$, we have

$$\mathbb{P}\left(|\widetilde{F}_\omega - \mathbb{E}[\widetilde{F}_\omega]| \geq \beta\right) = \mathbb{P}\left(|\widetilde{F}_\omega - F_{a^\star(p),\omega}| \geq \beta\right) \geq 1 - 2e^{-2q\beta^2}. \tag{6.6}$$

Then, by employing a union bound and the equation above, we have

$$\mathbb{P}\left(\|\widetilde{F} - F_{a^\star(p)}\|_\infty \leq \beta\right) = \mathbb{P}\left(\bigcap_{\omega \in \Omega} \left\{|\widetilde{F}_\omega - F_{a^\star(p),\omega}| \leq \beta\right\}\right) \geq 1 - 2me^{-2q\beta^2} \geq 1 - \lambda,$$

where the last inequality holds by definition of q. □

6.2.2 The `Try-Cover` procedure

In this section, we present key component of Algorithm 6.1, which is the `Try-Cover` procedure (Algorithm 6.3). It builds a cover $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$ of $\mathcal{P}$ made of approximate best-response regions for the meta-actions in the current set $\mathcal{D}$. The approximate best-response region $\mathcal{L}_d$ for a meta-action $d \in \mathcal{D}$ must be such that:

(i) all the best-response regions $\mathcal{P}_a$ of actions $a \in A(d)$ associated with d are contained in $\mathcal{L}_d$; and

(ii) for every contract in $\mathcal{L}_d$ there must be an action $a \in A(d)$ that is induced as an "approximate best response" by that contract.

Notice that working with approximate best-response regions is unavoidable since the algorithm has only access to estimates of the (true) distributions over outcomes induced by agent's actions.

The core idea of the algorithm is to progressively build two polytopes $\mathcal{U}_d$ and $\mathcal{L}_d$—called, respectively, *upper bound* and *lower bound*—for each meta-action $d \in \mathcal{D}$. During the execution, the upper bound $\mathcal{U}_d$ is continuously shrunk in such a way that a suitable approximate best-response region for d is guaranteed to be contained in $\mathcal{U}_d$, while the lower bound $\mathcal{L}_d$ is progressively expanded so that it is contained in such a region. The algorithm may terminate in two different ways. The first one is when $\mathcal{L}_d = \mathcal{U}_d$ for every $d \in \mathcal{D}$. In such a case, the lower bounds $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$ constitutes a cover of $\mathcal{P}$ made of suitable approximate best-response regions for the meta-actions in $\mathcal{D}$, and, thus, it is returned by the algorithm.

The second way of terminating occurs any time a call to the `Action-Oracle` procedure is *not* able to safely map the agent's best response under the contract given as input to a meta-action (*i.e.*, `Action-Oracle` returns $\perp$). In such a case, the algorithm gives back control to Algorithm 6.1 by returning $\varnothing$, and the latter in turn re-starts the `Try-Cover` procedure from scratch since the last call to `Action-Oracle` has updated $\mathcal{D}$. This happens in Lines 7, 17, and 25. The ultimate goal of Algorithm 6.3 is to reach termination with $\mathcal{L}_d = \mathcal{U}_d$ for every $d \in \mathcal{D}$. Intuitively, the algorithm tries to "close the gap" between the lower bound $\mathcal{L}_d$ and the upper bound $\mathcal{U}_d$ for each $d \in \mathcal{D}$ by discovering suitable *approximate halfspaces* whose intersections define the desired approximate best-response regions. Such halfspaces are found in Line 24 by means of the `Find-HS` procedure (Algorithm 6.4), whose description and analysis is deferred to Section 6.2.3. Intuitively, such a procedure searches for an hyperplane that defines a suitable approximate halfspace by performing a binary search (with parameter η) on the line connecting two given contracts, by calling `Action-Oracle` on the contract representing the middle point of the line at each iteration.

The halfspaces that define the approximate best-response regions computed by Algorithm 6.3 intuitively identify areas of $\mathcal{P}$ in which one meta-action is "supposedly better" than another one. In particular, Algorithm 6.3 uses the variable $\widetilde{H}_{ij}$ to store the approximate halfspace in which $d_i \in \mathcal{D}$ is "supposedly better" than $d_j \in \mathcal{D}$, in the sense that, for every contract in such a halfspace, each action associated with d_i provides (approximately) higher utility to the agent than all the actions associated with d_j. Then, the approximate best-response region for $d_i \in \mathcal{D}$

is built by intersecting a suitably-defined group of $\widetilde{H}_{ij}$, for some $d_j \in \mathcal{D}$ with $j \neq i$. To ease the construction of the approximate halfspaces in the `Find-HS` procedure, Algorithm 6.3 also keeps some variables $\Delta\widetilde{c}_{ij}$, which represent estimates of the difference between the costs $c(d_i)$ and $c(d_j)$ of d_i and d_j, respectively. These are needed to compute the intercepts of the separating hyperplanes defining $\widetilde{H}_{ij}$.

Algorithm 6.3 `Try-Cover`

1: $\mathcal{L}_d \leftarrow \varnothing, \mathcal{U}_d \leftarrow \mathcal{P}$ for all $d \in \mathcal{D}$
2: $\mathcal{D}_d \leftarrow \{d\}$ for all $d \in \mathcal{D}$
3: $\widetilde{H}_{ij} \leftarrow \varnothing$,
4: $\Delta\widetilde{c}_{ij} \leftarrow 0$ for all $d_i, d_j \in \mathcal{D}$
5: Randomly take a contract $p \in \mathcal{P}$
6: $d \leftarrow$ `Action-Oracle`(p)
7: **if** $d = \perp$ **then return** $\varnothing$
8: $\mathcal{L}_d \leftarrow \{p\}, \mathcal{C} \leftarrow \{d\}$
9: **while** $\mathcal{C} \neq \varnothing$ **do**
10: Take any $d_i \in \mathcal{C}$
11: **while** $\mathcal{L}_{d_i} \neq \mathcal{U}_{d_i}$ **do**
12: $V_{d_i} \leftarrow V(\mathcal{U}_{d_i})$ ▷ Set of vertices of $\mathcal{U}_{d_i}$
13: $\mathcal{L}_{d_i} \leftarrow \mathcal{L}_{d_i} \cap \mathcal{U}_{d_i}$
14: **do**
15: Take any $p \in V_{d_i}$
16: $d_j \leftarrow$ `Action-Oracle`(p)
17: **if** $d_j = \perp$ **then return** $\varnothing$
18: **if** $\mathcal{L}_{d_j} = \varnothing$ **then**
19: $\mathcal{L}_{d_j} \leftarrow \{p\}, \mathcal{C} \leftarrow \mathcal{C} \cup \{d_j\}$
20: **end if**
21: **if** $d_j \in \mathcal{D}_{d_i}$ **then**
22: $\mathcal{L}_{d_i} \leftarrow \mathrm{co}(\mathcal{L}_{d_i}, p)$
23: **else**
24: $(\widetilde{H}, d_k) \leftarrow$ `Find-HS`(d_i, p)
25: **if** $d_k = \perp$ **then return** $\varnothing$
26: $\widetilde{H}_{ik} \leftarrow \widetilde{H}$
27: $\mathcal{U}_{d_i} \leftarrow \mathcal{U}_{d_i} \cap \widetilde{H}_{ik}$
28: $\mathcal{D}_{d_i} \leftarrow \mathcal{D}_{d_i} \cup \{d_k\}$
29: **break**
30: **end if**
31: $V_{d_i} \leftarrow V_{d_i} \setminus \{p\}$
32: **while** $V_{d_i} \neq \varnothing \wedge d_j \in \mathcal{D}_{d_i}$
33: **end while**
34: $\mathcal{C} \leftarrow \mathcal{C} \setminus \{d_i\}$
35: **end while**
36: **return** $\{\mathcal{L}_d\}_{d \in \mathcal{D}}$

Next, we describe the procedure used by Algorithm 6.3 to reach the desired termination. In the following, for clarity of exposition, we omit all the cases in which the algorithm ends prematurely after a call to `Action-Oracle`. Algorithm 6.3 works by tracking all the $d \in \mathcal{D}$ that still need to be processed, *i.e.*, those such that $\mathcal{L}_d \neq \mathcal{U}_d$, into a set $\mathcal{C}$. At the beginning of the algorithm (Lines 1–8), all the vari-

ables are properly initialized. In particular, all the upper bounds are initialized to the set $\mathcal{P}$, while all the lower bounds are set to $\varnothing$. The set $\mathcal{C}$ is initialized to contain a $d \in \mathcal{D}$ obtained by calling `Action-Oracle` for any contract $p \in \mathcal{P}$, with the lower bound $\mathcal{L}_d$ being updated to the singleton $\{p\}$.

Moreover, Algorithm 6.3 also maintains some subsets $\mathcal{D}_d \subseteq \mathcal{D}$ of meta-actions, one for each $d \in \mathcal{D}$. For any $d_i \in \mathcal{D}$, the set $\mathcal{D}_{d_i}$ is updated so as to contain all the $d_j \in \mathcal{D}$ such that the halfspace $\widetilde{H}_{ij}$ has already been computed. Each set $\mathcal{D}_d$ is initialized to be equal to $\{d\}$, as this is useful to simplify the pseudocode of the algorithm. The main loop of the algorithm (Line 9) iterates over the meta-actions in $\mathcal{C}$ until such a set becomes empty. For every $d_i \in \mathcal{C}$, the algorithm tries to "close the gap" between the lower bound $\mathcal{L}_{d_i}$ and the upper bound $\mathcal{U}_{d_i}$ by using the loop in Line 11. In particular, the algorithm does so by iterating over all the vertices $V(\mathcal{U}_{d_i})$ of the polytope defining the upper bound $\mathcal{U}_{d_i}$ (see Line 12). For every $p \in V(\mathcal{U}_{d_i})$, the algorithm first calls `Action-Oracle` with p as input. Then:

- If the returned d_j belongs to $\mathcal{D}_{d_i}$, then the algorithm takes the convex hull between the current lower bound $\mathcal{L}_{d_i}$ and p as a new lower bound for d_i. This may happen when either $d_j = d_i$ (recall that $d_i \in \mathcal{D}_{d_i}$ by construction) or $d_j \neq d_i$. Intuitively, in the former case the lower bound can be "safely expanded" to match the upper bound at the currently-considered vertex p, while in the latter case such "matching" the upper bound may introduce additional errors in the definition of $\mathcal{L}_{d_i}$. Nevertheless, such an operation is done in both cases, since this *not* hinders the guarantees of the algorithm, as we formally show in Lemma 6.9. Notice that handling the case $d_j \neq d_i$ as in Algorithm 6.3 is crucial to avoid that multiple versions of the approximate halfspace $\widetilde{H}_{ij}$ are created during the execution of the algorithm.

- If the returned d_j does *not* belong to $\mathcal{D}_{d_i}$, the algorithm calls the `Find-HS` procedure to find a new approximate halfspace. Whenever the procedure is successful, it returns an approximate halfspace $\widetilde{H}$ that identifies an area of $\mathcal{P}$ in which d_i is "supposedly better" than another meta-action $d_k \in \mathcal{D} \setminus \mathcal{D}_{d_i}$, which is returned by the `Find-HS` procedure as well. Then, the returned halfspace $\widetilde{H}$ is copied into the variable $\widetilde{H}_{ik}$ (Line 26), and the upper bound $\mathcal{U}_{d_i}$ is intersected with the latter (Line 27). Moreover, d_k is added to $\mathcal{D}_{d_i}$ (Line 28), in order to record that the halfspace $\widetilde{H}_{ik}$ has been found. After that, the loop over vertexes is re-started, as the upper bound has been updated.

Whenever the loop in Line 11 terminates, $\mathcal{L}_{d_i} = \mathcal{U}_{d_i}$ for the current meta-action d_i, and, thus, d_i is removed from $\mathcal{C}$. Moreover, if a call to `Action-Oracle` returns a meta-action $d_j \in \mathcal{D}$ such that $d_j \notin \mathcal{C}$ and $\mathcal{L}_{d_j} = \varnothing$, then d_j is added to $\mathcal{C}$ and $\mathcal{L}_{d_j}$ is set to $\{p\}$, where $p \in \mathcal{P}$ is the contract given to `Action-Oracle` (see Lines 18–19). This ensures that all the meta-actions are eventually considered.

Before introducing the main properties guaranteed by Algorithm 6.3, we discuss Definition 6.5 for a given set of meta-actions $\mathcal{D}$. This definition associates to each meta-action d the set of contracts in which the agent's utility—computed using the empirical distribution over outcomes returned by the `Action-Oracle` procedure together with the cost of a meta-action introduced in Definition 6.4—is

greater than or equal to the utility computed with the same quantities for all the remaining meta-actions in $\mathcal{D}$. Formally, we have:

Definition 6.5. *Given a set of meta-actions $\mathcal{D}$, we let $\mathcal{P}_{d_i}(\mathcal{D}) \subseteq \mathcal{P}$ be the set defined as follows:*

$$\mathcal{P}_{d_i}(\mathcal{D}) \coloneqq \left\{ p \in \mathcal{P} \mid \sum_{\omega \in \Omega} \widetilde{F}_{d_i,\omega} p_\omega - c(d_i) \geq \sum_{\omega \in \Omega} \widetilde{F}_{d_j,\omega} p_\omega - c(d_j) \;\; \forall d_j \in \mathcal{D} \right\}.$$

We now formalize a relaxation of the boundary hyperplanes defining the regions $\mathcal{P}_{d_i}(\mathcal{D})$, parametrised by $y > 0$. Formally:

Definition 6.6. *Given $y \geq 0$, a set of meta-actions $\mathcal{D}$ and a dictionary $\mathcal{F}$ of empirical distributions computed by means of Algorithm 6.2, for every pair of meta-actions $d_i, d_j \in \mathcal{D}$, we let $H^y_{ij} \subseteq \mathcal{P}$ be the set of contracts defined as follows:*

$$H^y_{ij} \coloneqq \left\{ p \in \mathcal{P} \mid \sum_{\omega \in \Omega} \widetilde{F}_{d_i,\omega} p_\omega - c(d_i) \geq \sum_{\omega \in \Omega} \widetilde{F}_{d_j,\omega} p_\omega - c(d_j) - y \right\}.$$

Furthermore, we let $H_{ij} \coloneqq H^0_{ij}$.

It is important to notice that Definition 6.5 can equivalently be formulated by means of Definition 6.6. More specifically, given a set of meta-actions $\mathcal{D}$, for each $d_i \in \mathcal{D}$ we have

$$\mathcal{P}_{d_i}(\mathcal{D}) \coloneqq \bigcap_{d_j \in \mathcal{D}} H_{ij}.$$

As a further step, we introduce two useful lemmas. Lemma 6.6 shows that, for any set of meta-actions $\mathcal{D}$, the union of the sets $\mathcal{P}_d(\mathcal{D})$ over all $d \in \mathcal{D}$ coincides with the set $\mathcal{P}$. Conversely, Lemma 6.7 shows that, under the clean event, each set $\mathcal{P}_d(\mathcal{D})$ is contained in the upper bounds $\mathcal{U}_d$ computed by the `Try-Cover` procedure.

Lemma 6.6. *Given a set of meta-actions $\mathcal{D}$, it always holds $\bigcup_{d \in \mathcal{D}} \mathcal{P}_d(\mathcal{D}) = \mathcal{P}$.*

Proof. The lemma follows observing that, for each $p \in \mathcal{P}$, there exits $d_i \in \mathcal{D}$ such that:

$$\sum_{\omega \in \Omega} \widetilde{F}_{d_i,\omega} p_\omega - c(d_i) \geq \sum_{\omega \in \Omega} \widetilde{F}_{d_j,\omega} p_\omega - c(d_j) \;\; \forall d_j \in \mathcal{D}.$$

This is due to the fact that the cardinality of $\mathcal{D}$ is always ensured to be greater than or equal to one. Therefore, for each contract $p \in \mathcal{P}$, there exists a meta-action d such that $p \in \mathcal{P}_d(\mathcal{D})$, thus ensuring that

$$\bigcup_{d \in \mathcal{D}} \mathcal{P}_d(\mathcal{D}) = \mathcal{P}.$$

This concludes the proof. □

Lemma 6.7. *Given a set of meta-actions $\mathcal{D}$, under the event $\mathcal{E}_\epsilon$, it always holds $\mathcal{P}_d(\mathcal{D}) \subseteq \mathcal{U}_d$ for each $d \in \mathcal{D}$.*

Proof. To prove the lemma we observe that, for any meta-action $d_i \in \mathcal{D}$, the following inclusions hold:

$$\mathcal{U}_{d_i} = \bigcap_{d_j \in \mathcal{D}_{d_i}} \widetilde{H}_{ij} \supseteq \bigcap_{d_j \in \mathcal{D}_{d_i}} H_{ij} \supseteq \bigcap_{d_j \in \mathcal{D}} H_{ij} = \mathcal{P}_{d_i}(\mathcal{D}).$$

The first inclusion holds thanks to the definition of the halfspace $\widetilde{H}_{ij}$ and employing Lemma 6.12, which entails under the event $\mathcal{E}_\epsilon$. The second inclusion holds because $\mathcal{D}_{d_i}$ is a subset of $\mathcal{D}$ for each $d_i \in \mathcal{D}$. Finally, the last equality holds because of the definition of $\mathcal{P}_{d_i}(\mathcal{D})$. □

Next, we prove two crucial properties that are satisfied by Algorithm 6.3 whenever it returns $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$. Intuitively, the first property, stated in Lemma 6.8, shows that Algorithm 6.3 terminates with a correct cover of $\mathcal{P}$. This follows from the fact that, at the end of the algorithm, $\bigcup_{d\in\mathcal{D}} \mathcal{U}_d = \mathcal{P}$ and $\mathcal{L}_d = \mathcal{U}_d$ for every $d \in \mathcal{D}$. Formally:

Lemma 6.8. *Given a set of meta-actions $\mathcal{D}$, under the event $\mathcal{E}_\epsilon$, if Algorithm 6.3 returns $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$, then $\bigcup_{d\in\mathcal{D}} \mathcal{L}_d = \mathcal{P}$.*

Proof. As a first step, we observe that if Algorithm 6.3 returns $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$, then the set $\mathcal{D}$ has not been updated during its execution, and hence we must have $\mathcal{L}_d = \mathcal{U}_d$ for all $d \in \mathcal{D}$. Moreover, under the event $\mathcal{E}_\epsilon$, Lemma 6.6 together with Lemma 6.7 implies the following inclusion:

$$\bigcup_{d_i\in\mathcal{D}} \mathcal{U}_{d_i} \supseteq \bigcup_{d_i\in\mathcal{D}} \mathcal{P}_{d_i}(\mathcal{D}) = \mathcal{P}.$$

Then, by putting it all together we get:

$$\bigcup_{d_i\in\mathcal{D}} \mathcal{L}_{d_i} = \bigcup_{d_i\in\mathcal{D}} \mathcal{U}_{d_i} = \bigcup_{d_i\in\mathcal{D}} \mathcal{P}_i(\mathcal{D}) = \mathcal{P},$$

concluding the proof. □

The second property guaranteed by Algorithm 6.3 when it returns $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$ is stated in Lemma 6.9, which shows that this collection of sets defines a cover of $\mathcal{P}$ consisting of suitable approximate best-response regions for the meta-actions $\mathcal{D}$. Indeed, for every meta-action $d \in \mathcal{D}$ and contract $p \in \mathcal{L}_d$, playing any $a \in A(d)$ yields a utility for the agent that is only slightly smaller than the one obtained by playing the (true) best response $a^\star(p)$.

Lemma 6.9. *Given a set of meta-actions $\mathcal{D}$, under the event $\mathcal{E}_\epsilon$, when Algorithm 6.3 returns $\{\mathcal{L}_d\}_{d\in\mathcal{D}}$, for every meta-action $d \in \mathcal{D}$, contract $p \in \mathcal{L}_d$, and action $a' \in A(d)$, there exists a $\gamma \geq 0$ that depends polynomially on m, n, ϵ, and B such that:*

$$\sum_{\omega\in\Omega} F_{a',\omega}\, p_\omega - c_{a'} \geq \sum_{\omega\in\Omega} F_{a,\omega}\, p_\omega - c_a - \gamma \quad \forall a \in \mathcal{A}.$$

Proof. In order to prove the lemma, we rely on the crucial observation that, for any vertex $p \in V(\mathcal{L}_{d_i})$ of the lower bound of a meta-action $d_i \in \mathcal{D}$, the `Action-Oracle` procedure called by Algorithm 6.3 with p as input either returned d_i or another meta-action $d_j \in \mathcal{D}$ such that $p \in \widetilde{H}_{ij}$ with $d_j \in \mathcal{D}_{d_i}$. First, we consider the case in which the meta-action returned by `Action-Oracle` in the vertex p is equal to d_i. In such a case, $a^\star(p) \in A(d_i)$ thanks to Lemma 6.1 and the following holds:

$$\sum_{\omega \in \Omega} \widetilde{F}_{d_i,\omega}\, p_\omega - c(d_i) \geq \sum_{\omega \in \Omega} F_{a^\star(p),\omega}\, p_\omega - c_{a^\star(p)} - 9B\epsilon mn,$$

by means of Lemma 6.4. Next, we consider the case in which the meta-action returned by Algorithm 6.2 is equal to d_j with p that belongs to $\widetilde{H}_{ij}$. In such a case the following inequalities hold:

$$\begin{aligned}\sum_{\omega \in \Omega} \widetilde{F}_{d_i,\omega}\, p_\omega - c(d_i) &\geq \sum_{\omega \in \Omega} \widetilde{F}_{d_j,\omega}\, p_\omega - c(d_j) - y \\ &\geq \sum_{\omega \in \Omega} F_{a^\star(p),\omega}\, p_\omega - c_{a^\star(p)} - 9B\epsilon mn - y,\end{aligned}$$

where the first inequality follows by Lemma 6.12 that guarantees that p belongs to $\widetilde{H}_{ij} \subseteq H^y_{ij}$ with $y = 18B\epsilon mn^2 + 2n\eta\sqrt{m}$, while the second inequality holds because of Lemma 6.4, since $a^\star(p) \in A(d_j)$ thanks to Lemma 6.1 .

Finally, by putting together the inequalities for the two cases considered above and employing Lemma 6.4, we can conclude that, for every vertex $p \in V(\mathcal{L}_d)$ of the lower bound of a meta-action $d \in \mathcal{D}$, it holds:

$$\begin{aligned}\sum_{\omega \in \Omega} F_{a,\omega}\, p_\omega - c_a &\geq \sum_{\omega \in \Omega} \widetilde{F}_{d,\omega}\, p_\omega - c(d) - 9B\epsilon mn \\ &\geq \sum_{\omega \in \Omega} F_{a^\star(p),\omega}\, p_\omega - c_{a^\star(p)} - \gamma, \qquad (6.7)\end{aligned}$$

for each action $a \in A(d)$ by setting $\gamma \coloneqq 27B\epsilon mn^2 + 2n\eta\sqrt{m}$.

Moreover, observing that each lower bound $\mathcal{L}_d$ is a convex polytope, we can employ the Carathéodory's theorem to decompose each contract $p \in \mathcal{L}_d$ as a convex combination of the vertices of $\mathcal{L}_d$. Formally:

$$\sum_{p' \in V(\mathcal{L}_d)} \alpha(p')\, p'_\omega = p_\omega \quad \forall \omega \in \Omega, \qquad (6.8)$$

where $\alpha(p') \geq 0$ is the weight given to vertex $p' \in V(\mathcal{L}_d)$, so that it holds

$$\sum_{p' \in V(\mathcal{L}_d)} \alpha(p') = 1.$$

Finally, for every $p \in \mathcal{L}_d$ and action $a \in A(d)$ we have:

$$\begin{aligned}
\sum_{\omega \in \Omega} F_{a^\star(p),\omega}\, p_\omega - c_{a^\star(p)} &= \sum_{\omega \in \Omega} F_{a^\star(p),\omega} \left(\sum_{p' \in V(\mathcal{L}_d)} \alpha(p')\, p'_\omega \right) - c_{a^\star(p)} \\
&= \sum_{p' \in V(\mathcal{L}_d)} \alpha(p') \left(\sum_{\omega \in \Omega} F_{a^\star(p),\omega}\, p'_\omega - c_{a^\star(p)} \right) \\
&\leq \sum_{p' \in V(\mathcal{L}_d)} \alpha(p') \left(\sum_{\omega \in \Omega} F_{a,\omega}\, p'_\omega - c_a + \gamma \right) \\
&= \sum_{\omega \in \Omega} F_{a,\omega} \left(\sum_{p' \in V(\mathcal{L}_d)} \alpha(p')\, p'_\omega \right) - c_a + \gamma \\
&= \sum_{\omega \in \Omega} F_{a,\omega}\, p_\omega - c_a + \gamma,
\end{aligned}$$

where the first and the last equalities hold thanks to Equation (6.8), while the second and the third equalities hold since $\sum_{p' \in V(\mathcal{L}_d)} \alpha(p') = 1$. Finally, the inequality holds thanks to Equation (6.7) by setting $\gamma := 27B\epsilon mn^2 + 2n\eta\sqrt{m}$. □

Finally, we bound the number of rounds required by Algorithm 6.3 as follows.

Lemma 6.10. *Given a set of meta-actions $\mathcal{D}$, Algorithm 6.3 requires at most*

$$\mathcal{O}\left(|\mathcal{D}|^2 q \left(\log\left({}^{Bm}/_{\eta}\right) + \binom{m + |\mathcal{D}| + 1}{m} \right) \right)$$

rounds.

Proof. As a first step, we observe that Algorithm 6.3 terminates in a finite number of rounds. By the way in which Algorithm 6.3 intersects the upper bounds with the halfspaces computed with the help of the `Find-HS` procedure, the algorithm terminates with $\mathcal{L}_d = \mathcal{U}_d$ for all $d \in \mathcal{D}$ after a finite number of rounds as long as, for each meta-action $d_i \in \mathcal{D}$, the halfspaces $\widetilde{H}_{ij}$ with $d_j \in \mathcal{D}$ are computed at most once. It is easy to see that this is indeed the case. Specifically, for every meta-actions $d_i \in \mathcal{D}$, Algorithm 6.4 is called to build the halfspace $\widetilde{H}_{ij}$ only when Algorithm 6.2 returns d_j with $d_j \notin \mathcal{D}_{d_i}$ for a vertex $p \in V(\mathcal{U}_{d_i})$ of the upper bound $\mathcal{U}_{d_i}$. If the halfspace has already been computed, then $d_j \in \mathcal{D}_{d_i}$ by the way in which the set $\mathcal{D}_{d_i}$ is updated. As a result, if Algorithm 6.2 called on a vertex of the upper bound $\mathcal{U}_{d_i}$ returns the meta-action $d_j \in \mathcal{D}_{d_i}$, then Algorithm 6.3 does *not* compute the halfspace again.

For every $d_i \in \mathcal{D}$, the number of vertices of the upper bound $\mathcal{U}_{d_i}$ is at most $\binom{m+|\mathcal{D}|+1}{m} = \mathcal{O}(m^{|\mathcal{D}|+1})$, since the halfspaces defining the polytope $\mathcal{U}_{d_i}$ are a subset of the $m+1$ halfspaces defining $\mathcal{P}$ and the halfspaces $\widetilde{H}_{ij}$ with $d_j \in \mathcal{D}$. The number of halfspaces $\widetilde{H}_{ij}$ is at most $|\mathcal{D}|$ for every meta-action $d_i \in \mathcal{D}$.

Consequently, since each vertex lies at the intersection of at most m linearly independent hyperplanes, the total number of vertices of the upper bound $\mathcal{U}_{d_i}$ is bounded by $\binom{m+|\mathcal{D}|+1}{m} = \mathcal{O}(m^{|\mathcal{D}|+1})$, for every $d_i \in \mathcal{D}$

We also observe that, for every $d_i \in \mathcal{D}$, the while loop in Line 10 terminates with at most $V(\mathcal{U}_{d_i}) = \binom{m+|\mathcal{D}|+1}{m} = \mathcal{O}(m^{|\mathcal{D}|+1})$ iterations. This is because, during each iteration, either the algorithm finds a new halfspace or it exits from the loop. At each iteration of the loop, the algorithm invokes Algorithm 6.2, which requires q rounds. Moreover, finding a new halfspace requires a number of rounds of the order of $\mathcal{O}\left(q \log\left({}^{B m}/_{\eta}\right)\right)$, as stated in Lemma 6.13. Therefore, the total number of rounds required by the execution of the while loop in Line 10 is at most $\mathcal{O}\left(q\left(\log\left({}^{B m}/_{\eta}\right) + \binom{m+|\mathcal{D}|+1}{m}\right)\right)$.

Let us also notice that, during each iteration of the while loop in Line 11, either the algorithm finds a new halfspace or it exits from the loop. This is because, if no halfspace is computed, the algorithm does *not* update the boundaries of $\mathcal{U}_{d_i}$, meaning that the meta-action implemented in each vertex of $\mathcal{L}_{d_i}$ is either d_i or some d_j belonging to $\mathcal{D}_{d_i}$. Moreover, since the number of halfspaces $\widetilde{H}_{ij}$ is bounded by $|\mathcal{D}|$ for each $\mathcal{U}_{d_i}$, the while loop in Line 11 terminates in at most $|\mathcal{D}|$ steps. As a result, the number of rounds required by the execution of the while loop in Line 11 is of the order of $\mathcal{O}\left(|\mathcal{D}| q\left(\log({}^{B m}/_{\eta}) + \binom{m+|\mathcal{D}|+1}{m}\right)\right)$, being the while loop in Lines 10 nested within the one in Line 11.

Finally, we observe that the while loop in Line 3 iterates over the set of meta-actions actions $\mathcal{D}$. Therefore, the total number of rounds required to execute the entire algorithm is of the order of

$$\mathcal{O}\left(|\mathcal{D}|^2 q\left(\log\left({}^{B m}/_{\eta}\right) + \binom{m+|\mathcal{D}|+1}{m}\right)\right),$$

which concludes the proof. □

Lemma 6.10 follows from the observation that the main cost, in terms of number of rounds, incurred by Algorithm 6.3 is to check all the vertexes of the upper bounds $\mathcal{U}_d$. The number of such vertexes can be bound by $\binom{|\mathcal{D}|+m+1}{m}$, thanks to the fact that the set $\mathcal{P}$ being covered by Algorithm 6.3 has $m+1$ vertexes. Notice that, using $\mathcal{P}$ instead of $[0, B]^m$ is necessary, since the latter has a number vertexes which is exponential in m. Nevertheless, Algorithm 6.1 returns a contract $p \in [0, B]^m$ by means of the `Find-Contract` procedure, which we are going to describe in the following subsection.

6.2.3 The `Find-HS` procedure

In this section, we describe in details the `Find-HS` procedure (Algorithm 6.4). The procedure takes as inputs a meta-action $d_i \in \mathcal{D}$ and a contract $p \in \mathcal{P}$, and it tries to find one of the approximate halfspaces defining a suitable approximate best-response region for the meta-action d_i. It may terminate either with a pair $(\widetilde{H}, d_k)$ such that $\widetilde{H}$ is an approximate halfspace in which d_i is "supposedly better" than the meta-action $d_k \in \mathcal{D}$ or with a pair $(\varnothing, \perp)$, whenever a call to `Action-Oracle`

returned $\perp$. Let us recall that, for ease of presentation, we assume that Algorithm 6.4 has access to all the variables declared in the Try-Cover procedure (Algorithm 6.3), namely $\mathcal{L}_d$, $\mathcal{U}_d$, $\mathcal{D}_d$, $\widetilde{H}_{ij}$, and $\Delta\widetilde{c}_{ij}$. Algorithm 6.4 works by performing a binary search on the line segment connecting p with any contract in the (current) lower bound $\mathcal{L}_{d_i}$ of d_i (computed by the Try-Cover procedure). At each iteration of the search, by letting $p^1, p^2 \in \mathcal{P}$ be the two extremes of the (current) line segment, the algorithm calls the Action-Oracle procedure on the contract $p' \in \mathcal{P}$ defined as the middle point of the segment. Then, if the procedure returns a meta-action $d \in \mathcal{D}$ that belongs to $\mathcal{D}_{d_i}$ (meaning that the approximate halfspace $\widetilde{H}_{ij}$ has already been computed), the algorithm sets p^1 to p', while it sets p^2 to p' otherwise. The binary search terminates when the length of the line segment is below a suitable threshold $\eta \geq 0$.

After the search has terminated, the algorithm computes an approximate halfspace. First, the algorithm computes the estimate $\Delta\widetilde{c}_{ik}$ of the cost difference between the meta-actions d_i and d_k, where the latter is the last meta-action returned by Action-Oracle that does *not* belong to the set $\mathcal{D}_{d_i}$. Such an estimate is computed using the two empirical distributions $\widetilde{F}_{d_j}$ and $\widetilde{F}_{d_k}$ that the Action-Oracle procedure associated with the meta-actions $d_j \in \mathcal{D}_{d_i}$ and d_k, respectively. In particular, $\Delta\widetilde{c}_{ik}$ is computed as the sum of $\Delta\widetilde{c}_{ij}$—the estimate of the cost difference between meta-actions d_i and d_j that has been computed during a previous call to Algorithm 6.4—and an estimate of the cost difference between the meta-actions d_j and d_k, which can be computed by using the middle point of the line segment found by the binary search procedure (see Lines 23 and 25).

Then, the desired approximate halfspace is the one defined by the hyperplane passing through the middle point $p' \in \mathcal{P}$ of the line segment computed by the binary search with coefficients given by the $(m+1)$-dimensional vector $(\widetilde{F}_{d_i}^\top - \widetilde{F}_{d_k}^\top, \Delta\widetilde{c}_{ik} - y)$ (see Line 27), where $y \geq 0$ is a suitably-defined value chosen so as to ensure that the approximate best-response regions satisfy the desired properties (see Lemma 6.12).

Notice that Algorithm6.4 also needs some additional elements in order to ensure that the (calling) Try-Cover procedure is properly working. In particular, any time the Action-Oracle procedure returns $\perp$, Algorithm 6.4 terminates with $(\varnothing, \perp)$ as a return value (Line 9). This is needed to force the termination of the Try-Cover procedure (see Line 25 in Algorithm 6.3). Moreover, any time the Action-Oracle procedure returns a meta-action $d \notin \mathcal{C}$ such that $\mathcal{L}_d = \varnothing$, Algorithm 6.4 adds such a meta-action to $\mathcal{C}$ and it initializes its lower bound to the singleton $\{p'\}$, where p' is the contract given as input to Action-Oracle (see Lines 10–11). This is needed to ensure that all the meta-actions in $\mathcal{D}$ are eventually considered by the (calling) Try-Cover procedure.

We now prove some technical lemmas related to Algorithm 6.4. Lemma 6.11 provides a bound on the gap between the cost difference $\Delta\widetilde{c}_{ik}$ estimated by Algorithm 6.4 and the "true" cost difference between the meta-actions d_i and d_k (see Defintion 6.4). Lemma 6.12 shows that the approximate halfspace computed by the algorithm is always included in the halfspace introduced in Definition 6.6 for a suitably-defined value of the parameter $y \geq 0$. Finally, Lemma 6.13 provides a bound on the number of rounds required by the algorithm in order to terminate.

Algorithm 6.4 `Find-HS`

Require: d_i, p
1: $p^1 \leftarrow$ any contract in $\mathcal{L}_{d_i}$
2: $p^2 \leftarrow p$
3: $d_j \leftarrow$ `Action-Oracle`(p^1)
4: $d_k \leftarrow$ `Action-Oracle`(p^2)
5: $y \leftarrow 18B\epsilon mn^2 + 2n\eta\sqrt{m}$
6: **while** $\|p^1 - p^2\|_2 > \eta$ **do**
7: $\quad p'_\omega \leftarrow (p^1_\omega + p^2_\omega)/2$ for all $\omega \in \Omega$ ▷ Middle point of the current line segment
8: $\quad d \leftarrow$ `Action-Oracle`(p')
9: $\quad$ **if** $d = \perp$ **then return** $(\varnothing, \perp)$ ▷ Force termination of `Try-Cover`
10: $\quad$ **if** $\mathcal{L}_d = \varnothing$ **then** ▷ Add d to to-be-processed meta-actions in `Try-Cover`
11: $\quad\quad \mathcal{L}_d \leftarrow \{p'\}, \mathcal{C} \leftarrow \mathcal{C} \cup \{d\}$
12: $\quad$ **end if**
13: $\quad$ **if** $d \in \mathcal{D}_{d_i}$ **then**
14: $\quad\quad p^1_\omega \leftarrow p'_\omega$ for all $\omega \in \Omega$
15: $\quad\quad d_j \leftarrow d$
16: $\quad$ **else**
17: $\quad\quad p^2_\omega \leftarrow p'_\omega$ for all $\omega \in \Omega$
18: $\quad\quad d_k \leftarrow d$
19: $\quad$ **end if**
20: **end while**
21: $p'_\omega \leftarrow (p^1_\omega + p^2_\omega)/2$ for all $\omega \in \Omega$
22: **if** $d_j = d_i$ **then** ▷ The approximate halfspace $\widetilde{H}_{ik}$ is the first one for $\mathcal{L}_{d_i}$
23: $\quad \Delta\widetilde{c}_{ik} \leftarrow \sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega$
24: **else** ▷ The approximate halfspace $\widetilde{H}_{ij}$ has already been computed
25: $\quad \Delta\widetilde{c}_{ik} \leftarrow \Delta\widetilde{c}_{ij} + \sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega$
26: **end if**
27: $\widetilde{H} \coloneqq \left\{ p \in \mathbb{R}^m_+ \mid \sum_{\omega\in\Omega} \left(\widetilde{F}_{d_i,\omega} - \widetilde{F}_{d_k,\omega}\right) p_\omega \geq \Delta\widetilde{c}_{ik} - y \right\}$
28: **return** $(\widetilde{H}, d_k)$

Lemma 6.11. *Let $(\widetilde{H}, d_k)$ be the pair returned by Algorithm 6.4 when given as inputs $\mathcal{L}_{d_i}$ and $p \in \mathcal{P}$ for some meta-action $d_i \in \mathcal{D}$. Then, under the event $\mathcal{E}_\epsilon$, Algorithm 6.4 satisfies*

$$|\Delta\widetilde{c}_{ik} - c(d_i) + c(d_k)| \leq 18B\epsilon mn^2 + 2n\eta\sqrt{m}.$$

Proof. We start by proving that, after the execution of the loop at Line 6 in Algorithm 6.4, the following holds:

$$\left| \sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega - c(d_j) + c(d_k) \right| \leq 18B\epsilon mn + 2\eta\sqrt{m}, \tag{6.9}$$

where $d_j, d_k \in \mathcal{D}$ are the meta-actions resulting from the binary search procedure and $p' \in \mathcal{P}$ is the middle point point of the segment after the binary

search stopped. In order to prove Equation (6.9), we first let $a^1 := a^\star(p^1)$ and $a^2 := a^\star(p^2)$, where $p^1, p^2 \in \mathcal{P}$ are the two extremes of the line segment resulting from the binary search procedure. By Lemma 6.1, under the event $\mathcal{E}_\epsilon$, it holds that $a^1 \in A(d_j)$ and $a^2 \in A(d_k)$ by definition. Moreover, we let $p^* \in \mathcal{P}$ be the contract that belongs to the line segment connecting p^1 to p^2 and such that the agent is indifferent between actions a^1 and a^2, *i.e.*,

$$\sum_{\omega\in\Omega} \left(F_{a^1,\omega} - F_{a^2,\omega}\right) p^*_\omega = c_{a^1} - c_{a^2}.$$

Then, we can prove the following:

$$\begin{aligned}
&\left|\sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega - c(d_j) + c(d_k)\right| \\
&= \left|\sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega - c(d_j) + c_{a^1} - c_{a^1} + c_{a^2} - c_{a^2} + c(d_k)\right| \\
&\leq \left|\sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega - c_{a^1} + c_{a^2}\right| + 8B\epsilon mn \\
&= \left|\sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right) p'_\omega - \sum_{\omega\in\Omega} \left(F_{a^1,\omega} - F_{a^2,\omega}\right) p^*_\omega\right| + 8B\epsilon mn \\
&= \Bigg|\sum_{\omega\in\Omega} \left(\widetilde{F}_{d_j,\omega}\, p'_\omega + F_{a^1,\omega}\, p'_\omega - F_{a^1,\omega}\, p'_\omega - F_{a^1,\omega}\, p^*_\omega\right) \\
&\qquad + \sum_{\omega\in\Omega} \left(\widetilde{F}_{d_k,\omega}\, p'_\omega + F_{a^2,\omega}\, p'_\omega - F_{a^2,\omega}\, p'_\omega - F_{a^2,\omega}\, p^*_\omega\right)\Bigg| + 8B\epsilon mn \\
&\leq \left(\|\widetilde{F}_{d_j} - F_{a^1}\|_\infty + \|\widetilde{F}_{d_k} - F_{a^2}\|_\infty\right)\|p'\|_1 + (\|F_{a^1}\|_2 + \|F_{a^2}\|_2)\|p' - p^*\|_2 + 8B\epsilon mn \\
&\leq 18B\epsilon mn + 2\eta\sqrt{m}.
\end{aligned}$$

The first inequality above is a direct consequence of Lemma 6.3 and an application of the triangular inequality, since $a^1 \in A(d_j)$ and $a^2 \in A(d_k)$ under the event $\mathcal{E}_\epsilon$. The second inequality follows from Holder's inequality. The third inequality follows from Lemma 6.1 and Definition 6.3, noting that $\|p\|_1 \leq Bm$ and $\|F_a\| \leq \sqrt{m}$ for all $a \in \mathcal{A}$. Moreover, by the definitions of p' and p^*, and given the binary search performed by Algorithm 6.4, it is guaranteed that $\|p' - p^*\|_2 \leq \eta$.

Next, we prove that, after any call to Algorithm 6.4, the following holds:

$$|\Delta\widetilde{c}_{ik} - c(c_i) + c(d_k)| \leq |\mathcal{D}_{d_i}| \left(18B\epsilon mn + 2\eta\sqrt{m}\right),$$

where $d_i, d_k \in \mathcal{D}$ are the meta-actions defined by the binary search procedure in Algorithm 6.4. We prove the statement by induction on the calls to Algorithm 6.4 with the meta-action d_i as input. The base case is the first time Algorithm 6.4 is called with d_i as input. In that case, the statement is trivially satisfied

by Equation (6.9) and the fact that $\mathcal{D}_{d_i} = \{d_i\}$. Let us now consider a generic call to Algorithm 6.4 with the meta-action d_i as input. Then, we can write the following:

$$
\begin{aligned}
&|\Delta\widetilde{c}_{ik} - c(c_i) + c(d_k)| = \left|\Delta\widetilde{c}_{ij} + \sum_{\omega\in\Omega}\left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right)p'_\omega - c(d_i) + c(d_k)\right| \\
&\leq |\Delta\widetilde{c}_{ij} - c(d_i) + c(d_j)| + \left|\sum_{\omega\in\Omega}\left(\widetilde{F}_{d_j,\omega} - \widetilde{F}_{d_k,\omega}\right)p'_\omega - c(d_j) + c(d_k)\right| \\
&\leq (|\mathcal{D}_{d_i}| - 1)\left(18B\epsilon mn + 2\eta\sqrt{m}\right) + 18B\epsilon mn + 2\eta\sqrt{m} \\
&= |\mathcal{D}_{d_i}|\left(18B\epsilon mn + 2\eta\sqrt{m}\right),
\end{aligned}
$$

where the first inequality holds by applying the triangular inequality and the second one by using the inductive hypothesis. Finally, the proof is concluded observing that, under the clean event, $|\mathcal{D}_{d_i}| \leq n$. □

Lemma 6.12. *Let $(\widetilde{H}, d_k)$ be the pair returned by Algorithm 6.4 when given as inputs $\mathcal{L}_{d_i}$ and $p \in \mathcal{P}$ for some meta-action $d_i \in \mathcal{D}$. Then, under the event $\mathcal{E}_\epsilon$, it holds:*

$$H_{ik} \subseteq \widetilde{H} \subseteq H^y_{ik},$$

with $y = 18B\epsilon mn^2 + 2n\eta\sqrt{m}$.

Proof. We begin by showing that H_{ik} is a subset of $\widetilde{H}_{ik}$. Let p be a contract in H_{ik}. Then, by Definition 6.6, the following inequality holds:

$$\sum_{\omega\in\Omega}\widetilde{F}_{d_i,\omega}p_\omega - \sum_{\omega\in\Omega}\widetilde{F}_{d_k,\omega}p_\omega \geq \Delta\widetilde{c}_{ik} - y,$$

with $y = 18B\epsilon mn^2 + 2n\eta\sqrt{m}$ as prescribed by Lemma 6.11. This shows that $p \in \widetilde{H}_{ik}$, according to Definition 6.6. We now consider the case in which $p \in \widetilde{H}_{ik}$. Using the definition of $\widetilde{H}_{ik}$ and Lemma 6.11, we have:

$$\sum_{\omega\in\Omega}\widetilde{F}_{d_i,\omega}p_\omega - \sum_{\omega\in\Omega}\widetilde{F}_{d_k,\omega}p_\omega = \Delta\widetilde{c}_{ik} \geq c(d_i) - c(d_k) - y,$$

showing that $p \in H^y_{ik}$ with $y = 18B\epsilon mn^2 + 2n\eta\sqrt{m}$ and concluding the proof. □

Lemma 6.13. *The number of rounds required by Algorithm 6.4 to compute an approximate separating hyperplane is at most $\mathcal{O}\left(q \log\left({}^{Bm}/_{\eta}\right)\right)$.*

Proof. The lemma follows by observing that the number of rounds required by the binary search executed in the loop at Line 6 of Algorithm 6.4 is at most $\mathcal{O}(\log({}^{D}/_{\eta}))$, where D denotes the distance over which the binary search is performed. In our case, $D \leq \sqrt{2}Bm$, which is the maximum distance between two contracts in $\mathcal{P}$. Moreover, since the `Action-Oracle` algorithm is invoked at

each iteration, the number of rounds required by `Find-HS` is $\mathcal{O}(q \log(\sqrt{2}Bm/\eta))$, as the number of rounds required by `Action-Oracle` is bounded by q. □

6.2.4 The `Find-Contract` procedure

The `Find-Contract` procedure (Algorithm 6.5) finds a contract $p \in [0, B]^m$ that approximately maximizes the principal's expected utility over $[0, B]^m$ by using the cover $\{\mathcal{L}_d\}_{d \in \mathcal{D}}$ of $\mathcal{P}$ made by approximate best-response regions given as input (obtained by running `Try-Cover`).

Algorithm 6.5 `Find-Contract`

Require: $\{\mathcal{L}_d\}_{d \in \mathcal{D}}$
1: **while** $d \in \mathcal{D}$ **do**
2: $\quad p_d^\star \leftarrow \arg\max_{p \in [0,B]^m \cap \mathcal{L}_d} \sum_{\omega \in \Omega} \widetilde{F}_{d,\omega}\,(r_\omega - p_\omega)$
3: **end while**
4: $d^\star \leftarrow \arg\max_{d \in \mathcal{D}} \sum_{\omega \in \Omega} \widetilde{F}_{d,\omega}(r_\omega - p^\star_{d,\omega})$
5: $p^\star \leftarrow p^\star_{d^\star}$
6: $p_\omega \leftarrow (1 - \sqrt{\epsilon})p^\star_\omega + \sqrt{\epsilon} r_\omega$ for all $\omega \in \Omega$
7: **return** p

First, for every $d \in \mathcal{D}$, Algorithm 6.5 computes $p_d^\star$ which maximizes an empirical estimate of the principal's expected utility over the polytope $\mathcal{L}_d \cap [0, B]^m$. This is done in Line 2 by solving a *linear program* with constraints defined by the hyperplanes identifying $\mathcal{L}_d \cap [0, B]^m$ and objective function defined by the principal's expected utility when outcomes are generated by $\widetilde{F}_d$. Then, Algorithm 6.5 takes the best contract (according to the empirical distributions $\widetilde{F}_d$) among all the $p_d^\star$, which is the contract $p^\star$ defined in Line 5, and it returns a suitable convex combination of such a contract and a vector whose components are defined by principal's rewards (see Line 6). The following lemma formally proves the guarantees in terms of principal's expected utility provided by Algorithm 6.5:

Lemma 6.14. *Under the event $\mathcal{E}_\epsilon$, if $\{\mathcal{L}_d\}_{d \in \mathcal{D}}$ is a cover of $\mathcal{P}$ computed by* `Try-Cover`*, Algorithm 6.5 returns a contract $p \in [0, B]^m$ such that*

$$u^{\mathsf{P}}(p) \geq \max_{p' \in [0,B]^m} u^{\mathsf{P}}(p') - \rho.$$

Proof. In the following, we define $p^o \in [0, B]^m$ as the optimal contract within $[0, B]^m$, while we let

$$p^\ell := (1 - \sqrt{\gamma})p^o + \sqrt{\gamma} r,$$

where γ is defined as in Lemma 6.9. Additionally, we define $d^o \in \mathcal{D}$ as one of the meta-actions such that $p^o \in \mathcal{L}_{d^o}$. Similarly, we let $d^\ell \in \mathcal{D}$ be one of the meta-actions such that $p^\ell \in \mathcal{L}_{d^\ell}$. It is important to note that $p^\ell \in [0, B]^m$ since $\|r\|_\infty \leq 1$. Furthermore, Lemma 6.8 ensures that there exists at least one $d^\ell \in \mathcal{D}$ such that $p^\ell \in \mathcal{L}_{d^\ell}$. As a first step, we prove that, for each $a_i \in A(d^\ell)$, it holds:

$$
\begin{aligned}
\gamma &\geq \sum_{\omega \in \Omega} (F_{a^\star(p^\ell),\omega} - F_{a_i,\omega}) p^\ell_\omega + c_{a_i} - c_{a^\star(p^\ell)} \\
&\geq \sum_{\omega \in \Omega} (F_{a^\star(p^o),\omega} - F_{a_i,\omega}) p^\ell_\omega + c_{a_i} - c_{a^\star(p^o)} \\
&= \sum_{\omega \in \Omega} (F_{a^\star(p^o),\omega} - F_{a_i,\omega}) p^o_\omega + c_{a_i} - c_{a^\star(p^o)} + \sqrt{\gamma} \sum_{\omega \in \Omega} (F_{a^\star(p^o),\omega} - F_{a_i,\omega})(r_\omega - p^o_\omega) \\
&\geq \sqrt{\gamma} \sum_{\omega \in \Omega} (F_{a^\star(p^o),\omega} - F_{a_i,\omega})(r_\omega - p^o_\omega),
\end{aligned}
$$

where the first inequality holds because of Lemma 6.9 since $p^\ell \in \mathcal{L}_{d^\ell}$, while the second and the third inequalities hold because of the definition of best-response and the equality holds because of the definition of $p^\ell \in \mathcal{L}_{d^\ell}$.

Then, by rearranging the latter inequality, we can show that for each action $a_i \in A(d^\ell)$ the following holds:

$$
\sum_{\omega \in \Omega} F_{a_i,\omega}(r_\omega - p^o_\omega) \geq \sum_{\omega \in \Omega} F_{a^\star(p^o),\omega}(r_\omega - p^o_\omega) - \sqrt{\gamma}. \tag{6.10}
$$

Furthermore, for each action $a_i \in A(d^\ell)$, we have that:

$$
\begin{aligned}
\sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^\ell_\omega\right) &= \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - ((1-\sqrt{\gamma}) p^o_\omega + \sqrt{\gamma} r_\omega)\right) \\
&= \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^o_\omega\right) - \sqrt{\gamma} \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^o_\omega\right) \\
&\geq \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^o_\omega\right) - \sqrt{\gamma} \\
&\geq \sum_{\omega \in \Omega} F_{a^\star(p^o),\omega}(r_\omega - p^o_\omega) - 2\sqrt{\gamma},
\end{aligned}
$$

where the first equality holds because of the definition of $p^\ell \in \mathcal{L}_{d_\ell}$, while the first inequality holds since $\|r\|_\infty \leq 1$ and the latter inequality because of Equation (6.10). Putting all together, we get:

$$
\sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^\ell_\omega\right) \geq \sum_{\omega \in \Omega} F_{a^\star(p^o),\omega}(r_\omega - p^o_\omega) - 2\sqrt{\gamma} \tag{6.11}
$$

for each $a_i \in A(d^\ell)$ with $p^\ell \in \mathcal{L}_{d^\ell}$.

Now, we show that the principal's utility in the contract returned by Algorithm 6.5 is close to the optimal one. To do that we let $\{\widetilde{F}_d\}_{d \in \mathcal{D}}$ be the set of empirical distributions employed by Algorithm 6.5. Furthermore, we let $p^\star \in \mathcal{L}_{d^\star}$ be the contract computed in Line 5 of Algorithm 6.5. Analogously, we let

$$
p^f \coloneqq (1-\sqrt{\gamma}) p^\star + \sqrt{\gamma} r
$$

be the final contract the principal commits to. Then, for each $a_i \in A(d^\star)$, we have:

$$\begin{aligned} u^{\mathsf{P}}(p^f) &\geq \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^\star_\omega\right) - 2\sqrt{\gamma} \\ &\geq \sum_{\omega \in \Omega} \widetilde{F}_{d^\star,\omega} \left(r_\omega - p^\star_\omega\right) - 2\sqrt{\gamma} - 5\epsilon mn, \end{aligned}$$

where the first inequality follows from Proposition 2.4 by Dutting et al. [2021] and $u(p^\star) \leq 1$, while the second inequality holds by means of Definition 6.3 as $a_i \in A(d^\star)$. Analogously, for each $a_i \in A(d^\ell)$, we have:

$$\begin{aligned} \sum_{\omega \in \Omega} \widetilde{F}_{d^\star,\omega} \left(r_\omega - p^\star_\omega\right) &\geq \sum_{\omega \in \Omega} \widetilde{F}_{d^\ell,\omega} \left(r_\omega - p^\ell_\omega\right) \\ &\geq \sum_{\omega \in \Omega} F_{a_i,\omega} \left(r_\omega - p^\ell_\omega\right) - 5\epsilon mn \\ &\geq \max_{p' \in [0,B]^m} u^{\mathsf{P}}(p') - 2\sqrt{\gamma} - 5\epsilon mn, \end{aligned}$$

where the first inequality holds because of the optimality of $p^\star$, the second inequality holds because of Definition 6.3 since $a_i \in A(d^\ell)$, while the third inequality holds because of Equation (6.11). Finally, by putting all together we get:

$$\begin{aligned} u^{\mathsf{P}}(p^f) &\geq \max_{p' \in [0,B]^m} u^{\mathsf{P}}(p') - 4\sqrt{27B\epsilon mn^2 + 2n\eta\sqrt{m}} - 10\epsilon mn \\ &\geq \max_{p' \in [0,B]^m} u^{\mathsf{P}}(p') - 32\sqrt{B\epsilon m^2 n^2}, \end{aligned}$$

where we employ the definition of γ as prescribed by Lemma 6.9. Using the definitions of ϵ and η presented in Algorithm 6.1, the proof is concluded. □

The main challenge in proving Lemma 6.14 is that, for the contract $p^\star$ computed in Line 5, the agent's best response may *not* be associated with any meta-action in $\mathcal{D}$, namely $a^\star(p^\star) \notin A(d)$ for every $d \in \mathcal{D}$. Nevertheless, by means of Lemma 6.9, we can show that $a^\star(p^\star)$ is an approximate best response to $p^\star$. Moreover, the algorithm returns the contract defined in Line 6, *i.e.*, a convex combination of $p^\star$ and the principal's reward vector. Intuitively, paying the agent based on the principal's rewards aligns the agent's interests with those of the principal. This converts the approximate incentive compatibility of $a^\star(p^\star)$ for the contract $p^\star$ into a loss in terms of principal's expected utility.

6.2.5 Putting it all together

We conclude the chapter by combining the above results to establish the guarantees of Algorithm 6.1. First, we observe that the number of calls to the `Try-Cover` procedure is at most $2n$. Moreover, by Lemma 6.10 and by definition of q and η, the number of rounds required by each call to `Try-Cover` is at most $\widetilde{\mathcal{O}}(m^n \cdot \mathcal{I} \cdot {}^1\!/\!_{\rho^4} \log({}^1\!/\!_\delta))$ under the event $\mathcal{E}_\epsilon$, where $\mathcal{I}$ is a term that depends polynomially in

m, n, and B. Finally, by Lemma 6.14 the contract returned by Algorithm 6.1—the result of a call to the `Find-Contract` procedure—has expected utility at most ρ less than the best contract in $[0, B]^m$, while the probability of the clean event $\mathcal{E}_\epsilon$ can be bounded below by means of a concentration argument. All the observations above allow us to state the following main result.

Theorem 6.2. *Given $\rho \in (0, 1)$, $\delta \in (0, 1)$, and $B \geq 1$ as inputs, with probability at least $1 - \delta$ the* `Discover-and-Cover` *algorithm (Algorithm 6.1) is guaranteed to return a contract $p \in [0, B]^m$ such that $u(p) \geq \max_{p' \in [0,B]^m} u(p') - \rho$ in at most*

$$\widetilde{\mathcal{O}}(m^n \cdot \mathcal{I} \cdot 1/\rho^4 \log(1/\delta))$$

rounds, where $\mathcal{I}$ is a term that depends polynomially in m, n, and B.

Proof. First, we observe that Lemma 6.14 returns a contract $p \in \mathcal{P}$ such that, under the event $\mathcal{E}_\epsilon$, it holds

$$u^{\mathsf{P}}(p) \geq \max_{p' \in [0,B]^m} u^{\mathsf{P}}(p') - \rho.$$

Indeed, under $\mathcal{E}_\epsilon$ and thanks to Lemma 6.2, there exists an execution of the `Try-Cover` procedure that does not return $\varnothing$, and thus produces a collection of lower bounds which is given as input to Algorithm 6.5.

We prove that $\mathbb{P}(\mathcal{E}_\epsilon) \geq 1 - \delta$. To this end, we bound the number of times the `Discover-and-Cover` procedure invokes Algorithm 6.2. By Lemma 6.10, the number of times Algorithm 6.2 is called by the `Try-Cover` procedure when receiving as input a set $\mathcal{D}$ is bounded by

$$|\mathcal{D}|^2 \left(\log\left(\frac{2Bm}{\eta}\right) + \binom{m + |\mathcal{D}| + 1}{m} \right).$$

Moreover, the `Try-Cover` procedure is invoked at most $2n$ times during the execution of the `Discover-and-Cover` procedure. Consequently, the number of times Algorithm 6.2 is invoked is bounded by

$$2n|\mathcal{D}|^2 \left(\log\left(\tfrac{2Bm}{\eta}\right) + \binom{m + |\mathcal{D}| + 1}{m} \right).$$

By applying a union bound over all the invocations of Algorithm 6.2, and noting that each time it returns an empirical distribution that is within ϵ in the $\|\cdot\|_\infty$ norm of the true distribution with probability at least $1 - \alpha$, we conclude that the event $\mathcal{E}_\epsilon$ occurs with probability at least

$$\mathbb{P}(\mathcal{E}_\epsilon) \geq 1 - 8\alpha n^3 \left(\log\left(\frac{2Bm}{\eta}\right) + \binom{m + 2n + 1}{m} \right),$$

since, during the execution of Algorithm 6.1, it always holds that $|\mathcal{D}| \leq 2n$. This is because each time Algorithm 6.3 returns $\varnothing$, the size of $\mathcal{D}$ is subsequently increased by at most one, and the number of times Algorithm 6.3 is invoked

during the execution of Algorithm 6.1 is at most $2n$, according to the design of Algorithm 6.1. As a result, by setting:

$$\alpha := \frac{\delta}{8n^3\left(\log\left(2Bm/\eta\right) + \binom{m+2n+1}{m}\right)}, \tag{6.12}$$

with η defined as in Algorithm 6.1, we have $\mathbb{P}(\mathcal{E}_\epsilon) \geq 1 - \delta$.

We bound the number of samples employed by Algorithm 6.1. Under the event $\mathcal{E}_\epsilon$, the cardinality of $\mathcal{D}$ is at most n. Thus, with the same reasoning as above, we have that the number of samples used by Algorithm 6.1 can be bounded as:

$$\mathcal{O}\left(qn^3\left(\log\left(2Bm/\eta\right) + \binom{m+n+1}{m}\right)\right),$$

where q is the number of samples used by the `Action-Oracle` procedure. Therefore, by definition of the parameters α, ϵ, η, and q , the total number of rounds required by Algorithm 6.1 to return a ρ-optimal solution with probability at least $1 - \delta$ is at most:

$$\widetilde{\mathcal{O}}\left(m^{n+1}\frac{B^2m^4n^8}{\rho^4}\log\left(\frac{1}{\delta}\right)\right),$$

which concludes the proof. □

Notice that the number of rounds required by Algorithm 6.1 is polynomial in the instance size (including the number of outcomes m) when the number of agent's actions n is constant.

6.3 A no-regret algorithm

In this section, we show that our `Discover-and-Cover` algorithm can be exploited to derive a no-regret algorithm for the related online learning setting in which the principal aims at maximizing their cumulative utility. In such a setting, the principal and the agent interact repeatedly over a given number of rounds T.

Algorithm 6.6 No-regret algorithm

Require: $\delta \in (0, 1)$, $B \geq 1$
1: Set ρ as in proof of Theorem 6.3
2: **for** $t = 1, \ldots, T$ **do**
3: **if** Algorithm 6.1 *not* terminated yet **then**
4: $p^t \leftarrow p \in \mathcal{P}$ prescribed by Algorithm 6.1
5: **else**
6: $p^t \leftarrow p \in [0, B]^m$ returned by Algorithm 6.1
7: **end if**
8: **end for**

At each $t = 1, \ldots, T$, the principal commits to a contract $p^t \in \mathbb{R}^m_+$, the agent plays a best response $a^\star(p^t)$, and the principal observes the outcome $\omega^t \sim F_{a^\star(p^t)}$ with reward r_{ω^t}. Then, the performance in terms of cumulative expected utility by

employing the contracts $\{p^t\}_{t=1}^T$ is measured by the *cumulative* (*Stackelberg*) *regret*

$$R^T := T \cdot \max_{p \in [0,B]^m} u^{\mathsf{P}}(p) - \sum_{t=1}^{T} u^{\mathsf{P}}(p^t),$$

As shown in Section 6.2, Algorithm 6.1 learns an approximately-optimal bounded contract with high probability by using the number of rounds prescribed by Theorem 6.2. Thus, by exploiting Algorithm 6.1, it is possible to design an *explore-then-commit* algorithm ensuring sublinear regret R^T with high probability; see Algorithm 6.6.

Theorem 6.3. *Given $\delta \in (0,1)$, Algorithm 6.6 achieves*

$$R^T \leq \widetilde{\mathcal{O}}\left(m^n \cdot \mathcal{I} \cdot \log(1/\delta) \cdot T^{4/5}\right)$$

with probability at least $1-\delta$, where $\mathcal{I}$ is a term depending polynomially on m, n, and B.

Proof. Thanks to Theorem 6.2, we know that by employing an appropriate number of rounds, the solution returned by Algorithm 6.1 differs by at most ρ from that of an optimal bounded contract with probability at least $1-\delta$. Furthermore, we notice that the per-round regret suffered by Algorithm 6.6 is bounded by $mB+1$ during the execution of Algorithm 6.1, and it is at most ρ for the remaining rounds. Formally, we have that:

$$R_T \leq \widetilde{\mathcal{O}}\left(m^n \frac{B^3 m^5 n^8}{\rho^4} \log\left(\frac{1}{\delta}\right) + T\rho\right).$$

Thus, by setting $\rho = m^{5/4} B^{3/5} m n^{8/5} T^{-1/5}$ as input to Algorithm 6.1, with probability at least $1-\delta$ the cumulative regret is bounded by:

$$R_T \leq \widetilde{\mathcal{O}}\left(m^n B^{3/5} n^{8/5} \log\left(\frac{1}{\delta}\right) T^{4/5}\right),$$

concluding the proof. □

Notice that, even for a small number of outcomes m (*i.e.*, any $m \geq 3$), our algorithm achieves better regret guarantees than those obtained by Zhu et al. [2023] in terms of the dependency on the number of rounds T. Specifically, Zhu et al. [2023] provide a $\widetilde{\mathcal{O}}(T^{1-1/(2m+1)})$ regret bound, which exhibits a very unpleasant dependency on the number of outcomes m at the exponent of T. Conversely, our algorithm always achieves a $\widetilde{\mathcal{O}}(T^{4/5})$ dependency on T, when the number of agent's actions n is small. This solves a problem left open in a paper by Zhu et al. [2023].

6.4 Related works

The work that is most related to this chapter is [Zhu et al., 2023], which investigate a setting very similar to ours, though from an online learning perspective. Zhu et al. [2023] study general hidden-action principal-agent problem instances in which the principal faces multiple agent's types. They provide a regret bound

of the order of $\widetilde{\mathcal{O}}(\sqrt{m} \cdot T^{1-1/(2m+1)})$ when the principal is restricted to contracts in $[0,1]^m$, where m is the number of outcomes. They also show that their regret bound can be improved to $\widetilde{\mathcal{O}}(T^{1-1/(m+2)})$ by making additional structural assumptions on the problem instances, including the *first-order stochastic dominance* (FOSD) condition. Moreover, Zhu et al. [2023] provide an (almost-matching) lower bound of $\Omega(T^{1-1/(m+2)})$ that holds even with a single agent's type, thus showing that the dependence on the number of outcomes m at the exponent of T is necessary in their setting. Notice that the regret bound by Zhu et al. [2023] is "close" to a linear dependence on T, even when m is very small. In contrast, in Section 6.3 we show how our algorithm can be exploited to achieve a regret bound whose dependence on T is of the order of $\widetilde{\mathcal{O}}(T^{4/5})$ (independent of m), when the number of agent's actions n is constant. Another work that is closely related to ours is the one by Ho et al. [2014], who focus on designing a no-regret algorithm for a particular repeated principal-agent problem. However, their approach relies heavily on stringent structural assumptions, such as the first-order stochastic dominance (FOSD) condition. Furthermore, Cohen et al. [2022] study a repeated principal-agent interaction with a *risk-averse* agent, providing a no-regret algorithm that relies on the FOSD condition too. More recently, Chen et al. [2024] developed an algorithm to learn approximately optimal contracts under the FOSD and the concavity of distribution function property (CDFP) assumptions.[2]

[2]Intuitively, FOSD implies that actions with higher costs for the agent have a higher probability of leading to favorable outcomes for the principal. Furthermore, an action $a_i \in \mathcal{A}$ satisfies the CDFP property if, for any two actions such that the cost of action a_i is a weighted average of their costs, the distribution over outcomes of action a_i stochastically dominates the weighted average of the outcome distributions of the two actions.

CHAPTER 7

Learning linear contracts

In this chapter, we study the problem of designing no-regret algorithms with respect to an optimal, utility-maximizing *linear contract*. These contracts represent a powerful and widely used subclass of bounded contracts, where the principal commits to pay a fixed fraction of the realized reward, and they arise naturally in many economic applications due to their simplicity and robustness.

To tackle this problem we introduce a new and more general learning framework, called *Bandits with Monotone Jumps* (BwMJ). This framework provides a unifying perspective for a broad class of online learning problems in which the learner's expected utility has a piecewise linear structure satisfying some monotonicity properties, and the learner receives noisy bandit feedback. Beyond linear contracts, we show that this framework also captures other important problems in microeconomic theory, such as posted-price auctions when the distribution of buyers' valuations has finite support, and first-price auctions under similar assumptions.

Our main result is an algorithm that achieves a regret bound of $\widetilde{O}(\sqrt{nT})$, where n denotes the number of "jumps" in the piecewise linear reward function of the learner and T the time horizon. This result directly implies the existence of a $\widetilde{O}(\sqrt{nT})$ no-regret algorithm for linear contracts, where n corresponds to the number of the agent's actions, thereby improving upon the $\mathcal{O}(T^{2/3})$ bound previously proposed by Zhu et al. [2023] in settings where n is smaller compared to $T^{1/3}$. In addition, we establish a matching lower bound (up to logarithmic factors), showing that this rate is tight in these regimes, thereby providing nearly optimal guarantees for this class of problems.

7.1 Bandits with monotone jumps

In this section, we formally introduce the *Bandit with Monotone Jumps* (BwMJ) problems addressed in this chapter. As we discuss in the following, BwMJ problems provide a general framework that captures several learning tasks arising from microeconomic models, such as posted-price auctions and algorithmic contract design in principal-agent problems.

7.1.1 Model

An instance of BwMJ is defined as a triplet $\mathcal{I} \coloneqq \big((\mathcal{A}_i)_{i\in[n]}, (\nu_i)_{i\in[n]}, \ell\big)$, where:

- $(\mathcal{A}_i)_{i\in[n]}$ is a collection of intervals s.t. $\mathcal{A}_i \coloneqq [\overline{\alpha}_i, \overline{\alpha}_{i+1})$ for every $i \in [n-1]$ and $\mathcal{A}_n \coloneqq [\overline{\alpha}_n, \overline{\alpha}_{n+1}]$, with $\overline{\alpha}_1 = 0$, $\overline{\alpha}_{n+1} = 1$, and $\overline{\alpha}_i < \overline{\alpha}_{i+1}$ for all $i \in [n]$.
- $(\nu_i)_{i\in[n]}$ is a collection of probability distributions in which each distribution ν_i has support $\mathrm{supp}(\nu_i) \subseteq [0,1]$ and expected value $\mu_i \in [0,1]$, with $\mu_i < \mu_{i+1}$ for all $i \in [n-1]$.[1]
- $\ell : [0,1] \to [0,1]$ is a strictly decreasing linear function.[2]

In a BwMJ problem, the learner only knows the linear function ℓ, while they do *not* know anything about the intervals $\mathcal{A}_i$ and their corresponding probability distributions ν_i, including their supports and expected values μ_i. Let us also remark that the terminology "monotone jumps" refers to the jump discontinuities characterizing the expected rewards of the learner, as described later.

The action space available to the learner is the interval $[0,1]$. Whenever the learner selects an action $\alpha \in [0,1]$, they receive as feedback a sample from a probability distribution ν_i, where $i \in [n]$ is the index of the interval containing α, namely $\alpha \in \mathcal{A}_i$. Thus, every time the learner selects an action belonging to some interval $\mathcal{A}_i$, they receive as feedback a sample from the same distribution ν_i. For ease of notation, we introduce a function $h : [0,1] \to [n]$ that takes $\alpha \in [0,1]$ as input and returns the index $i \in [n]$ such that $\alpha \in \mathcal{A}_i$. Formally, for every $\alpha \in [0,1]$, it holds that:

$$h(\alpha) \coloneqq \sum_{i\in[n]} \mathbb{1}\{\alpha \in \mathcal{A}_i\} \cdot i.$$

Notice that the function h is always well defined, as the intervals $(\mathcal{A}_i)_{i\in[n]}$ are disjoint by definition. After selecting an action $\alpha \in [0,1]$ and observing feedback $x \sim \nu_{h(\alpha)}$, the learner achieves a reward (or utility) equal to the product $\ell(\alpha) \cdot x$. We introduce a function $u : [0,1] \to \mathbb{R}$ to compactly denote the learner's expected reward as a function of their action $\alpha \in [0,1]$. Formally:

$$u(\alpha) \coloneqq \ell(\alpha)\, \mathbb{E}_{x\sim\nu_{h(\alpha)}}[x] = \ell(\alpha)\, \mu_{h(\alpha)}.$$

[1] Notice that, for ease of presentation, we focus on BwMJ instances in which $\mu_i < \mu_{i+1}$ for all $i \in [n-1]$. However, our results can be easily generalized to instances where $\mu_i > \mu_{i+1}$ for all $i \in [n-1]$.

[2] Notice that, since the sequence of expected values $(\mu_i)_{i\in[n]}$ is strictly increasing, we can focus w.l.o.g. on BwMJ instances in which $\ell : [0,1] \to [0,1]$ is a strictly decreasing linear function. Otherwise, as it will be clear later, the optimal action for the learner would trivially be $\alpha = 1$. Moreover, let us remark that our results can be extended to deal with BwMJ instances in which $\ell : [0,1] \to [0,1]$ belongs to a broader class of functions. However, since linear functions are sufficient to encompass the main applications presented in this chapter, we only deal with linear functions, for ease of presentation. Finally, the BwMJ framework is stated for increasing jumps and a decreasing linear function. This is without loss of generality: all results apply symmetrically to the case of decreasing jumps and an increasing linear function

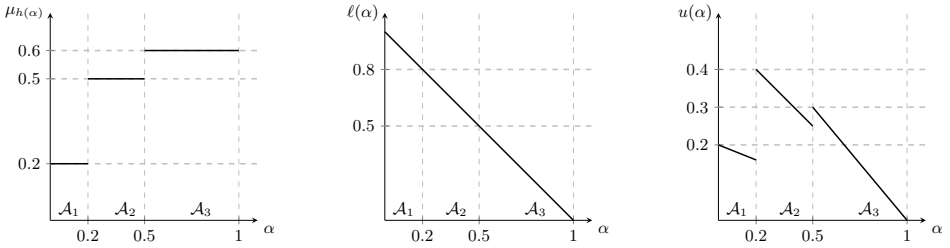

Figure 7.1: *Example of BwMJ instance.* (Left) *The expected values $\mu_{h(\alpha)}$ as a function of α, where $h(\alpha)$ is defined as the index i such that $\alpha \in \mathcal{A}_i$.* (Center) *The linear function $\ell(\alpha) \coloneqq 1 - \alpha$.* (Right) *The learner's expected reward $u(\alpha) \coloneqq \ell(\alpha)\mu_{h(\alpha)}$, which is a piecewise linear function over the action space $[0, 1]$.*

Notice that, even if the function $\ell : [0, 1] \to [0, 1]$ is linear, the learner's expected reward $u(\alpha)$ is in general a *piecewise linear* function, and, thus, it may be *discontinuous*. Indeed, the function $u(\alpha)$ is characterized by $n - 1$ jump discontinuity points, corresponding to actions defining the "meeting point" of two consecutive intervals $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$, where the value of $u(\alpha)$ abruptly "jumps" due to the fact that $\mu_{h(\alpha)}$ changes from μ_i to μ_{i+1}. We call such actions (*monotone*) *jumps* since, by assumption, the value of $\mu_{h(\alpha)}$ can only increase as α increases, given that $\mu_{i+1} - \mu_i > 0$. Moreover, we refer to $\mu_{i+1} - \mu_i$ as the *gap* of the jump at the "meeting point" of the intervals $\mathcal{A}_i$ and $\mathcal{A}_{i+1}$. Figure 7.1 graphically depicts an example of BwMJ instance.

7.1.2 Learner's policies and regret

In a BwMJ problem, the learner repeatedly interacts with the environment over $T \in \mathbb{N}_+$ rounds. At each round $t \in [T]$, the learner-environment interaction goes as follows:

1. The learner selects an action α_t from the interval $[0, 1]$.
2. Given the action selected by the agent, the environment samples $x_t \sim \nu_{h(\alpha_t)}$.
3. The learner observes x_t as feedback and achieves reward $\ell(\alpha_t) \cdot x_t$.

The learner's behavior over the T rounds is described by a (*deterministic*) *policy*, which is defined as a tuple $\pi \coloneqq (\pi_t)_{t\in[T]}$ whose components $\pi_t : \mathcal{H}_{t-1} \to [0, 1]$ encode how the learner selects actions at each round. Specifically, π_t is a mapping from the history of observations up to the preceding round $t - 1$, namely $H_{t-1} \coloneqq (\alpha_0, x_0, \alpha_1 \ldots, \alpha_{t-1}, x_{t-1}) \in \mathcal{H}_{t-1}$, to an action $\alpha_t = \pi_t(H_{t-1})$, where the set $\mathcal{H}_{t-1}$ contains all the possible histories of observations up to round $t - 1$.

The performance of a learner's policy $\pi \coloneqq (\pi_t)_{t\in[T]}$ over the T rounds is evaluated in terms of the (*cumulative*) *pseudo-regret* (henceforth sometimes called *regret* for short), which is defined as:

$$R_T(\pi) \coloneqq T \cdot \mathrm{OPT} - \mathbb{E}\left[\sum_{t\in[T]} u(\alpha_t)\right],$$

where the expectation is taken with respect to the randomness of the environment that generates the feedback received by the learner at each round. In the regret definition, OPT represents the learner's expected reward for an action that is *optimal in hindsight*. This is defined as $\text{OPT} := \max_{\alpha \in [0,1]} u(\alpha)$.[3] In the following, for any $\epsilon \in [0,1]$, we say that an action $\alpha \in [0,1]$ is ϵ-optimal if $u(\alpha) \geq \text{OPT} - \epsilon$. Furthermore, in the rest of this chapter, we sometimes omit the dependency on the policy from the regret, by simply writing $R_T(\pi)$ as R_T, whenever the policy π is clear from context.

The goal of the learner is to find a policy that minimizes the pseudo-regret. Ideally, the learner would like a policy achieving *no-regret*, meaning that the regret grows sublinearly in the number of rounds T, namely $R_T = o(T)$. This ensures that the per-round regret $\frac{R_T}{T}$ goes to zero as T grows.

We conclude the section with two observations related to the attainability of the learner's goal.

Why the jump monotonicity condition is needed? As a first observation, let us remark that, without assuming that the expected values μ_i constitute a monotone sequence, namely $\mu_i < \mu_{i+1}$ for all $i \in [n-1]$ (or equivalently, $\mu_i > \mu_{i+1}$ for all $i \in [n-1]$), it is *not* possible to attain no-regret. This is because it is always possible to construct a BwMJ instance $\mathcal{I} := \big((\mathcal{A}_i)_{i\in[n]}, (\nu_i)_{i\in[n]}, \ell\big)$ in which the learner can obtain a utility greater than zero only over an arbitrarily small interval. Indeed, it is sufficient to consider an instance in which there exists an index $i \in [n]$ such that $\mu_i = 1$ and $\mathcal{A}_i = [\alpha^\star, \alpha^\star + \epsilon)$ for $\alpha^\star \in (0,1)$ and an arbitrarily small $\epsilon > 0$, while $\mu_j = 0$ for all the other indexes $j \neq i$. Clearly, since the learner does *not* know $\alpha^\star$, the probability with which any algorithm selects an action belonging to $\mathcal{A}_i$ can be made arbitrarily close zero as ϵ goes to zero. This results in any algorithm suffering a pseudo-regret that grows linearly in the number of rounds T.

Why existing algorithms are not enough? As a second observation, let us remark that a naïve approach to achieve no-regret in BwMJ problems would be to exploit existing algorithms for continuum-armed bandit settings with one-sided *Lipschitz* rewards. Indeed, it is easy to check that, in BwMJ problems, the learner's expected reward function $u(\alpha)$ is always one-sided *Lipschitz*. As a result, it is possible to employ the algorithm by Kleinberg [2004] for continuum-armed bandits to achieve pseudo-regret of the order of $\widetilde{\mathcal{O}}(T^{2/3})$. This regret bound is optimal for the general bandit instances considered in [Kleinberg, 2004]. However, as we show in the rest of this chapter, in BwMJ problems it is possible to improve the regret bound when the number of discontinuities is sufficiently small, by employing an *ad hoc* approach that substantially differs from the one adopted by Kleinberg [2004], so as to attain a regret bound of the order of $\widetilde{\mathcal{O}}(\sqrt{nT})$.

7.1.3 No-regret learning in linear contracts

We now introduce the subclass of *linear contracts*, which simply transfer a fraction $\rho \in [0,1]$ of principal's rewards to the agent, namely $p_j = \rho r_j$ for every $j \in [m]$.

[3] Notice that the learner's expected reward *always admits a maximum* in a BwMJ instance. This follows from the fact that the linear function ℓ is decreasing and the intervals $\mathcal{A}_i$ are closed on the left. Thus, within each interval $\mathcal{A}_i$, the maximum exists and is attained at the left extreme of the interval. Since the union of all the intervals $\mathcal{A}_i$ gives the whole space of actions $[0,1]$, the learner's expected reward clearly admits a maximum.

Under a linear contract parametrized by $\rho \in [0,1]$, if the agent selects an action $i \in [n]$, then the principal's expected utility is equal to $(1-\rho)R_i$, while the agent's one is $\rho R_i - c_i$. After the principal commits to a linear contract $\rho \in [0,1]$, the agent chooses a *best response*, that is, an action maximizing their expected utility under ρ. With a slight abuse of notation, we denote it by $i^\star(\rho) \in [n]$. When the agent is indifferent among multiple actions, we assume that ties are broken in favor of the principal, as it is standard in the literature (see, *e.g.*, [Dütting et al., 2019]). Thus, the principal's expected utility is $u(\rho) \coloneqq (1-\rho)R_{i^\star(\rho)}$ as a function of $\rho \in [0,1]$.[4]

Figure 7.2 (*Left*) shows an example of principal's expected utility $u(\rho)$.

In the following, for the sake of presentation, we assume w.l.o.g. that agent's actions are ordered in increasing order of expected reward, so that $R_i < R_{i+1}$ for every $i \in [n-1]$. Moreover, we assume that all the actions are implementable by at least one linear contract, which means that, for every action $i \in [n]$, there exists some $\rho \in [0,1]$ such that $\rho R_i - c_i > \rho R_j - c_j$ for all $j \in [n]$ with $j \neq i$.

When the principal-agent interaction is repeated for $T \in \mathbb{N}_+$ rounds, at each round $t \in [T]$:

1. The principal commits to a linear contract $\rho_t \in [0,1]$.
2. The agent plays a best response $i^\star(\rho_t)$, which is *not* observed by the principal.
3. The principal observes the outcome $j \sim F_{i^\star(\rho_t)}$ realized by the agent's action.
4. The principal obtains utility $(1-\rho_t)r_j$, while the agent receives $\rho_t r_j - c_{i^\star(\rho_t)}$.

In such a repeated setting, a learning algorithm for the principal has to prescribe a linear contract to commit to at each round, without any knowledge about costs and probability distributions associated with agent's actions. The goal is to design a policy π whose regret

$$R_T(\pi) = T \cdot u(\rho^\star) - \mathbb{E}\left[\sum_{t\in[T]} u(\rho_t)\right]$$

grows sublinearly in the number of rounds T, where $\rho^\star \in \arg\max_{\rho\in[0,1]} u(\rho)$ denotes an optimal-in-hindsight linear contract (see, *e.g.*, [Zhu et al., 2023]). This is equivalent to achieving no-regret in a suitable BwMJ instance, which we describe in the following. First, let us observe that agent's best responses induce a partition of linear contracts. Indeed, by letting $\mathcal{B}_i \coloneqq \{\rho \in [0,1] \mid i^\star(\rho) = i\}$ be the set of contracts under which action $i \in [n]$ is a best response, it is easy to see that $\mathcal{B}_i = [\beta_i, \beta_{i+1})$ for every $i \in [n]$, for a suitable sequence $(\beta_i)_{i\in[n+1]}$ such that $\beta_i < \beta_{i+1}$ for all $i \in [n]$, $\beta_1 = 0$, and $\beta_{n+1} = 1$ (see, *e.g.*, [Dütting et al., 2019]).

Then, the BwMJ instance can be defined as follows.

- The intervals $(\mathcal{A}_i)_{i\in[n]}$ coincide with the sets $\mathcal{B}_i = [\beta_i, \beta_{i+1})$.
- The distributions $(\nu_i)_{i\in[n]}$ are discrete and supported over all the possible principal's rewards $(r_j)_{j\in[m]}$, with the probability that ν_i assigns to value r_j being equal to $F_{i,j}$.

[4] With a slight abuse of notation, we represent actions $\mathcal{A} = \{a_i\}_{i=1}^n$ and states $\Omega = \{\omega_j\}_{j=1}^m$ directly by their indices. That is, we write i instead of a_i and j instead of ω_j.

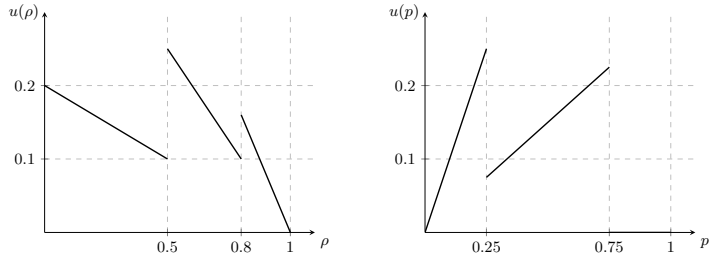

Figure 7.2: (Left) *Example of principal's expected utility by using linear contracts in a principal-agent problem.* (Right) *Example of seller's expected utility in a posted-price auction.*

- The linear function $\ell : [0,1] \to [0,1]$ is such that $\ell(\rho) = 1-\rho$ for every $\rho \in [0,1]$.

Finally, let us observe that the monotonicity of the jumps is guaranteed by the fact that, by definition, $\mu_i = \mathbb{E}_{j\sim F_i}[r_j] = R_i$, and the sequence of principal's expected rewards $(R_i)_{i\in[n]}$ is increasing.

7.2 Bandits with monotone jumps applied to auctions

In this section, we show that the BwMJ framework captures several learning tasks arising from auction models. In Section 7.2.1, we provide a connection with learning problems arising in *posted-price* auctions with few buyer's valuations. In Section 7.2.2, we provide a connection with learning problems arising in *first-price* auctions with finitely many bidders.

7.2.1 No-Regret in posted-price auctions

In a *posted-price* auction, a seller seeks to sell an item to a buyer, whose valuation for the item is sampled according to some unknown probability distribution (see, *e.g.*, the random model in [Kleinberg and Leighton, 2003]). Formally, the seller posts a price $p \in [0,1]$ for the item, while the buyer's valuation $v \in (0,1)$ for the item is sampled from a probability distribution $\mathcal{V}$, which has support $\mathrm{supp}(\mathcal{V}) \subseteq (0,1)$. Then, the buyer chooses to acquire the item only if their valuation v is greater than or equal to the price p proposed by the seller. Consequently, the seller's expected utility is equal to $u(p) := p \cdot \mathbb{P}_{v\sim\mathcal{V}}(v \geq p) = p \cdot (1 - F(p))$ for every $p \in [0,1]$, where $F : [0,1] \to [0,1]$ denotes a cumulative density function of $\mathcal{V}$ such that $F(x) := \mathbb{P}_{v\sim\mathcal{V}}(v < x)$ for all $x \in [0,1]$. In Figure 7.2 (*Right*) we report an example of seller's expected utility.

In the following, we assume that $\mathcal{V}$ is a discrete probability distribution such that $|\mathrm{supp}(\mathcal{V})| = n$ for some $n \in \mathbb{N}_+$, and we define the support of $\mathcal{V}$ as $\mathrm{supp}(\mathcal{V})$. When a posted-price auction is repeated for $T \in \mathbb{N}_+$ rounds, at each $t \in [T]$:

1. The seller proposes a price $p_t \in [0,1]$ to the buyer.
2. The buyer's valuation $v_t \sim \mathcal{V}$ is sampled according to the distribution $\mathcal{V}$.
3. If $v_t \geq p_t$, then the buyer buys the item, the seller gains p_t, and the buyer gets utility $v_t - p_t$. Otherwise, if $v_t < p_t$, both the seller and the buyer get utility zero.

Notice that, at the end of each round, the seller does *not* observe v_t, but only whether the item has been sold or not. In such a setting, the seller has *no* knowledge about the distribution $\mathcal{V}$. Thus, the learning task that they face is to minimize the regret with respect to the choice of always proposing an optimal-in-hindsight price, which is computed with knowledge of $\mathcal{V}$.

The learning task introduced above can be equivalently formulated in the framework of BwMJ problems. Specifically, it can be captured by means of a BwMJ instance defined as follows.

- There are $n+1$ intervals $(\mathcal{A}_i)_{i\in[n+1]}$ such that $\mathcal{A}_1 \coloneqq [v_0, v_1]$, with $v_0 \coloneqq 0$, and $\mathcal{A}_i \coloneqq (v_{i-1}, v_i]$ for all $i \in [n+1]$, with $v_{n+1} \coloneqq 1$. Intuitively, for all the prices in an interval $\mathcal{A}_i$, the item is sold with the same probability. Specifically, this is because, for all the valuations $v_j \in \mathrm{supp}(\mathcal{V})$ with $j \geq i$, the buyer buys the item, while, for all the other valuations, they do *not* buy.

- The probability distributions $(\nu_i)_{i\in[n+1]}$ are *Bernoulli* with parameters (*i.e.*, expected values) $\mu_i \coloneqq 1 - F(v_i)$. Let us remark that μ_i is the probability of selling the item when the posted price lies within the interval $\mathcal{A}_i$, as $F(v_i) = \mathbb{P}_{v\sim\mathcal{V}}(v < v_i)$ and $1 - F(v_i) = \mathbb{P}_{v\sim\mathcal{V}}(v \geq v_i)$.

- The linear function $\ell : [0,1] \to [0,1]$ is such that $\ell(p) = p$ for every $p \in [0,1]$.

7.2.2 No-regret in first-price auctions

In a *first-price* auction, a bidder with a private valuation $v \in [0,1]$ for the item being sold competes with other bidders in order to win the item, by placing a bid $b \in [0,1]$. The bidders simultaneously send their bids, and the one with the highest bid wins the item, paying their own bid. By letting $m \in [0,1]$ be the value of the highest bid submitted by the other bidders (*i.e.*, excluding bid b), the bidder wins the item if and only if $b \geq m$ (assuming ties are broken in their favor), and they pay b, thus achieving a utility of $v - b$. Otherwise, if $b < m$, the bidder does *not* win, and their utility is equal to zero. In the following, as it is common in the literature (see, *e.g.*, Han et al. [2024b]), we assume that the maximum bid submitted by the other bidders is sampled from a probability distribution $\mathcal{V}$, which is unknown to the bidder and satisfies $\mathrm{supp}(\mathcal{V}) \subseteq [0,1]$ with $|\mathrm{supp}(\mathcal{V})| = n$ for some $n \in \mathbb{N}_+$. Thus, it is easy to verify that the bidder's expected utility, as a function of the submitted bid $b \in [0,1]$, is equal to $u(b) \coloneqq (v - b) \cdot \mathbb{P}_{m\sim\mathcal{V}}(b \geq m)$.

When a bidder with valuation $v \in [0,1]$ repeats the auction for $T \in \mathbb{N}_+$ rounds, at each $t \in [T]$:[5]

1. The bidder chooses a bid $b_t \in [0,1]$.
2. The maximum bid of the other bidders $m_t \sim \mathcal{V}$ is sampled according to the distribution $\mathcal{V}$.
3. The bidder's utility is $v - b_t$ if $b_t \geq m_t$. Otherwise, if $b_t < m_t$, the bidder's utility is zero.

[5] Notice that we assume that the bidder's valuation is the same at every round, while, in most of the works in the literature Han et al. [2024b], Ai et al. [2022], it is assumed that it is sampled at every round according to some fixed probability distribution. However, let us remark that, the latter setting cannot be captured by a BwMJ problem.

The learning interaction described above corresponds to an equivalent one in a suitably-defined BwMJ instance. We let each interval $\mathcal{A}_i$ be the set of bids for which the probability of winning the auction remains the same. Furthermore, we let each distribution ν_i be a *Bernoulli* distribution whose expected value corresponds to the probability of winning the auction in the interval $\mathcal{A}_i$. Finally, we define the function $\ell(\alpha) = v - \alpha$.

7.3 A no-regret algorithm for the BwMJ framework

In this section, we present the main result of this chapter, which is a no-regret learning algorithm for BwMJ problems, whose regret guarantees are tight up to logarithmic factors. The section is organized as follows. Section 7.3.1 introduces the main procedure executed by the algorithm, while Section 7.3.2 and Section 7.3.3 focus on two core sub-procedures. Finally, Section 7.3.4 concludes by proving the regret guarantees attained by the algorithm.

7.3.1 Recursive jump identification with optimistic shrinking

The main procedure executed by the algorithm—called *Recursive Jump Identification with Optimistic Shrinking* (`RJI-OS` for short)—is provided in Algorithm 7.1. The procedure splits the T rounds into several epochs, with each epoch $j \in \mathbb{N}_+$ being further divided into three macro blocks.[6]

In the *first* macro block (Lines 2-5), Algorithm 7.1 attempts at identifying all the jump discontinuities whose corresponding gap $\mu_{i+1} - \mu_i$ is of the order of $\Delta_j := 1/2^j$. To do this, the algorithm invokes the `Find-Jumps` procedure (Algorithm 7.2), which employs a recursive binary-search-style approach. In the *second* macro block (Lines 7-14), by leveraging the information about jumps collected by `Find-Jumps`, Algorithm 7.1 computes an estimate of OPT for the current epoch j, denoted as $\text{OPT}(j)$. By using such an estimate and the information collected by `Find-Jumps`, in the *third* macro block (Lines 15-18), Algorithm 7.1 reduces the set of actions $\alpha \in [0, 1]$ to be considered during epoch $j + 1$. This is done by employing a suitably-designed procedure called `Optimistic-Shrink` (Algorithm 7.3), whose effects are crucial in order to achieve the desired regret guarantees.

Algorithm 7.1 keeps track of all the information it needs into two data structures. The first one is a collection of action intervals that the `Find-Jumps` procedure has to consider during the current epoch, denoted as $\mathcal{I}_j$ for epoch $j \in \mathbb{N}_+$. Such a collection contains the whole action space $[0, 1]$ during the first epoch, namely $\mathcal{I}_1 := \{[0, 1]\}$, and it is updated at the end of each epoch (in the third macro block) by the `Optimistic-Shrink` procedure. Notice that, during an epoch $j \in \mathbb{N}_+$, the set $\mathcal{I}_j$ may contain potentially non-adjacent action intervals, which do *not* necessarily cover the whole action space $[0, 1]$. In the second data structure, which is denoted by $\mathcal{T}_j$ for epoch $j \in \mathbb{N}_+$, the algorithm keeps track of all the information collected by the `Find-Jumps` procedure in the current epoch. Specifically, each element of $\mathcal{T}_j$ is a triplet specifying an action interval $I = [\alpha_1, \alpha_2]$ that has been returned by `Find-Jumps`, an estimate $\widehat{\mu}(\alpha_1)$ of the expected value

[6] Notice that, for the sake of presentation, in Algorithm 7.1 and all its sub-procedures we do *not* keep track of the current round $t \in [T]$. However, we assume that, whenever $t > T$, the execution of the algorithm is immediately stopped.

Algorithm 7.1 `Recursive Jump Identification with Optimistic Shrinking`

Require: Number of rounds $T \in \mathbb{N}_+$
1: Set $\mathcal{I}_1 \leftarrow \{[0,1]\}$
2: **for** $j \in \mathbb{N}_+$ **do**
3: $\quad\mathcal{T}_j \leftarrow \varnothing, \Delta_j \leftarrow 1/2^j$ ▷ First macro block
4: $\quad$**for** $I \in \mathcal{I}_j$ **do**
5: $\quad\quad\mathcal{T}_j \leftarrow \mathcal{T}_j \cup$ `Find-Jumps`$(I, \Delta_j, 1)$
6: $\quad$**end for**
7: $\quad$OPT$(j) \leftarrow 0, \mathcal{I}_{j+1} \leftarrow \varnothing$ ▷ Second macro block
8: $\quad$**for** $([\alpha_1, \alpha_2], \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$ **do**
9: $\quad\quad$**for** $\alpha \in \{\alpha_1, \alpha_2\}$ **do**
10: $\quad\quad\quad$**if** $\ell(\alpha) \cdot \widehat{\mu}(\alpha) >$ OPT(j) **then**
11: $\quad\quad\quad\quad$OPT$(j) \leftarrow \ell(\alpha) \cdot \widehat{\mu}(\alpha), \alpha_j^\star \leftarrow \alpha$
12: $\quad\quad\quad$**end if**
13: $\quad\quad$**end for**
14: $\quad$**end for**
15: $\quad$**for** $\tau = ([\alpha_1, \alpha_2], \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$ **do** ▷ Third macro block
16: $\quad\quad\mathcal{I}_{j+1} \leftarrow \mathcal{I}_{j+1} \cup$ `Optimistic-Shrink`$(\tau,$ OPT$(j), j)$
17: $\quad$**end for**
18: **end for**

$\mu_{h(\alpha_1)}$, and an estimate $\widehat{\mu}(\alpha_2)$ of $\mu_{h(\alpha_2)}$. Notice that $\mathcal{T}_j$ is re-initialized at the beginning of each epoch. Next, we provide more details about the three macro blocks in each epoch $j \in \mathbb{N}_+$ of Algorithm 7.1.

1. In the *first* macro block (Lines 2-5), Algorithm 7.1 iterates over the intervals in $\mathcal{I}_j$. For each interval, Algorithm 7.1 invokes the `Find-Jumps` procedure to identify, within the interval, some jump discontinuity whose corresponding gap is of the order of Δ_j. To do so, `Find-Jumps` recursively calls itself, each time halving the interval given as input. Specifically, given an interval $I = [\alpha_1, \alpha_2]$, `Find-Jumps` first plays both the extremes of the interval for a number of rounds proportional to $1/\Delta_j^2$. This allows to build suitable estimates $\widehat{\mu}(\alpha_1)$ and $\widehat{\mu}(\alpha_2)$ of $\mu_{h(\alpha_1)}$ and $\mu_{h(\alpha_2)}$, respectively. If the difference between such estimates is larger than Δ_j, then `Find-Jumps` recursively invokes itself on the two halves of I. Otherwise, it stops returning a triplet defined as $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2))$. All the triplets generated by the different executions of `Find-Jumps` are stored by Algorithm 7.1 in $\mathcal{T}_j$, in order to be used in the subsequent blocks.

2. In the *second* macro block (Lines 7-14), Algorithm 7.1 computes the current estimate OPT(j) of the optimal expected utility OPT, by using the estimates previously computed by `Find-Jumps`. Specifically, Algorithm 7.1 iterates over all the triplets $(I = [\alpha_1, \alpha_2], \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$, and, for each of them, it computes the estimated expected utility at both the extremes of I, as $\ell(\alpha) \cdot \widehat{\mu}(\alpha)$ for $\alpha \in \{\alpha_1, \alpha_2\}$. The maximum among all such values is taken as OPT(j), while the extreme point in which it is attained is stored in $\alpha_j^\star$.

3. In the *third* macro block (Lines 15-18), Algorithm 7.1 calls the `Optimistic-Shrink` procedure to identify a collection $\mathcal{I}_{j+1}$ of action intervals to be con-

sidered by the `Find-Jumps` procedure in the following epoch $j+1$. The main goal of this step is to ensure that, in each possible $\alpha \in [0,1]$ selected during epoch $j+1$, the learner's expected utility satisfies $u(\alpha) \geq \text{OPT} - \mathcal{O}(\Delta_j)$. In this way, the regret incurred by the algorithm during the execution of `Find-Jumps` in the first block of epoch $j+1$ is *not* too large. This is crucial to achieve the desired regret guarantees.

7.3.2 The `Find-Jumps` procedure

In this section, we present the `Find-Jumps` procedure, whose pseudocode is provided in Algorithm 7.2. At an high level, given an interval as input, the algorithm implements a recursive binary search in order to identify, within the interval, jump discontinuities whose associated gaps are proportional to a given Δ_j. Specifically, the algorithm takes as input a triplet (I, Δ_j, k), where $I = [\alpha_1, \alpha_2] \subseteq [0,1]$ is an action interval, $\Delta_j \in [0,1]$ is a jump gap considered by Algorithm 7.1 in epoch $j \in \mathbb{N}_+$, and $k \in \mathbb{N}_+$ is the depth reached in the tree of recursive calls made by `Find-Jumps`.

Before delving into the details of Algorithm 7.2, we make an observation that helps to better understand how it works. Given an interval $I = [\alpha_1, \alpha_2] \subseteq [0,1]$, by playing each α_i for $\mathcal{O}\left(1/\Delta_j^2\right)$ rounds, it is possible to compute some estimates $\widehat{\mu}(\alpha_i)$ whose distance from the true expected values $\mu_{h(\alpha_i)}$ is of the order of Δ_j, with high probability. Thus, if $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1)$ is larger than Δ_j, then, with high probability, there exists at least one jump discontinuity within the interval I. However, notice that this does *not* guarantee that there exists a *single* jump discontinuity with gap proportional to Δ_j. Indeed, there could be multiple jump discontinuities with smaller gaps, whose cumulative sum is proportional to Δ_j. Instead, if $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1)$ is smaller than Δ_j, then either there is no jump discontinuity in the interval I or a gap smaller than Δ_j is needed in order to identify one.

Algorithm 7.2 `Find-Jumps`

Require: A triplet (I, Δ_j, k) with $I = [\alpha_1, \alpha_2] \subseteq [0,1]$, $\Delta_j \in [0,1]$, and $k \in \mathbb{N}_+$

$\delta \leftarrow 1/T$
$N \leftarrow 8/\Delta_j^2 \log(4T/\delta)$
if $\alpha_2 - \alpha_1 \leq \frac{1}{T}$ **then**
    Play $\alpha_t := \alpha_2$ and observe feedback $x_t \sim \nu_{h(\alpha_t)}$ for N rounds
    Compute estimate $\widehat{\mu}(\alpha_2)$ using the observed feedback
    $\widehat{\mu}(\alpha_1) \leftarrow 0$
    return $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2))$
end if
for $i = 1, 2$ **do**
    Play $\alpha_t := \alpha_i$ and observe feedback $x_t \sim \nu_{h(\alpha_t)}$ for N rounds
    Compute estimate $\widehat{\mu}(\alpha_i)$ using the observed feedback
end for
if $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) \geq \Delta_j$ **then**
    return `Find-Jumps` $\left([\alpha_1, \frac{\alpha_1+\alpha_2}{2}], \Delta_j, k+1\right) \cup$ `Find-Jumps` $\left([\frac{\alpha_1+\alpha_2}{2}, \alpha_2], \Delta_j, k+1\right)$
else
    return $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2))$
end if

We now describe how Algorithm 7.2 works. This can be conceptually split into two cases, depending on whether the length of the interval $I = [\alpha_1, \alpha_2]$ given as input is greater than $1/T$ or not.

Case $\alpha_2 - \alpha_1 \leq 1/T$. In this case (see Lines 3-7), Algorithm 7.2 does *not* search for jump discontinuities within the interval in input, and it returns the interval itself without further halving it. Intuitively, this is because the interval is sufficiently small for the purposes of the algorithm. This represents the first base case of the recursive binary search implemented by Algorithm 7.2. Before returning, the algorithm plays for $N \propto 1/\Delta_j^2$ rounds the right extreme α_2 of the interval, in order to compute an estimate $\widehat{\mu}(\alpha_2)$ of $\mu_{h(\alpha_2)}$, while it artificially sets the estimate $\widehat{\mu}(\alpha_1)$ for the left extreme α_1 to zero. The reason why Algorithm 7.2 computes an estimate only for the right extreme of the interval is that, as it will become clear later on, this extreme provides an expected utility that is sufficiently close to the highest expected utility attainable over the whole interval. Finally, in this case the algorithm ends its execution with Line 7, by returning the triplet $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2))$.

Case $\alpha_2 - \alpha_1 > 1/T$. In this case (see Lines 9-16), Algorithm 7.2 searches for a jump discontinuity within the interval given as input. In order to do so, the algorithm first computes two estimates $\widehat{\mu}(\alpha_1)$ and $\widehat{\mu}(\alpha_2)$ of the expected values $\mu_{h(\alpha_1)}$ and $\mu_{h(\alpha_2)}$ at the extremes of the interval, by playing each α_i for a suitable number of rounds N proportional to $1/\Delta_j^2$. Then, depending on the value of the difference between such estimates, the algorithm proceeds its execution as follows.

1. If the difference $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1)$ is smaller than Δ_j, Algorithm 7.2 stops searching for jumps discontinuities within the input interval, and, thus, it does *not* invoke itself anymore (see Line 16). Indeed, in this case, either there is no jump discontinuity in the interval or a smaller value of Δ_j is needed in order to identify one, as we previously observed. This represents another base case in the recursive binary search implemented by Algorithm 7.2, in which the algorithm simply returns the triplet defined as $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2))$ (see Line 16).

2. If the difference $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1)$ is larger than Δ_j, Algorithm 7.2 halves the input interval I and recursively invokes itself with these two new subintervals as inputs (see Line 13). This is because, if the difference between the estimates is greater than Δ_j, then there could be a jump discontinuity whose gap is proportional to Δ_j within that interval, as previously observed.

Next, we provide some results related to Algorithm 7.2. In the following, for the ease of presentation, we introduce a *clean event* $\mathcal{E}$. This encodes all the situations in which the estimates computed by Algorithm 7.2 fall within a suitable confidence interval, in *every* execution of the algorithm.

Definition 7.1 (Clean event)**.** *We define $\mathcal{E}$ as the event in which, every time Algorithm 7.2 is called given as input a triplet (I, Δ_j, k) with $I = [\alpha_1, \alpha_2] \subseteq [0, 1]$, $\Delta_j \in [0, 1]$, and $k \in \mathbb{N}_+$, the estimates $\widehat{\mu}(\alpha_1)$ and $\widehat{\mu}(\alpha_2)$ computed by the algorithm satisfy the following conditions:*

$$\left|\widehat{\mu}(\alpha_i) - \mu_{h(\alpha_i)}\right| \leq \frac{\Delta_j}{4} \quad \forall i \in \{1, 2\}.$$

We now prove two lemmas that state two crucial properties that Algorithm 7.2 satisfies.

The first property ensures that, whenever Algorithm 7.2 stops recursively calling itself, then either the interval given as input is "sufficiently" tight or the jump discontinuities within the interval (if there are some) have gaps that are "small enough". Formally:

Lemma 7.1. *Suppose that Algorithm 7.2 is given as input a triplet (I, Δ_j, k), with $I = [\alpha_1, \alpha_2] \subseteq [0, 1]$, $\Delta_j \in [0, 1]$, and $k \in \mathbb{N}_+$. Under the clean event $\mathcal{E}$, if the algorithm does not recursively call itself, then one of the following two conditions holds:*

1. $\alpha_2 - \alpha_1 \leq \frac{1}{T}$.
2. $\mu_{h(\alpha)} - \mu_{h(\alpha')} \leq \frac{3}{2}\Delta_j$ *for all* $\alpha, \alpha' \in I$ *such that* $\alpha \geq \alpha'$.

Proof. To prove the lemma, we notice that when Algorithm 7.2 returns a tuple containing the same interval I it has received as input, either the condition $|I| \leq 1/T$ or the condition $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) < \Delta_j$ holds. This is due to the termination conditions specified in Algorithm 7.2.

In the following, we consider these two possible termination conditions and we show that if either one of them holds, then either (1) or (2) holds.

1. If $|I| \leq 1/T$, then condition (1) trivially holds.
2. If $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) < \Delta_j$, then for all $\alpha, \alpha' \in I$ with $\alpha \geq \alpha'$, under the event $\mathcal{E}$, we have:

$$\mu_{h(\alpha)} - \mu_{h(\alpha')} \leq \mu_{h(\alpha_2)} - \mu_{h(\alpha_1)} \leq \widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) + \frac{\Delta_j}{2} \leq \frac{3}{2}\Delta_j.$$

Where the first inequality holds because of monotonicity, the second inequality holds under the event $\mathcal{E}$ and the last inequality holds because $\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) < \Delta_j$ by hypothesis, and, thus condition (2) holds.

Thanks to the two points above, the proof is concluded. □

The second crucial property that Algorithm 7.2 ensures is that, whenever it receives as input an interval such that the distributions at its extremes coincide, and, thus, they have the same expected value, the algorithm does *not* invoke itself further. Such a property is crucial; otherwise, Algorithm 7.2 may perform too many unnecessary recursive calls, resulting in a large regret. Formally:

Lemma 7.2. *Suppose that Algorithm 7.2 is given as input a triplet (I, Δ_j, k) with $I = [\alpha_1, \alpha_2] \subseteq [0, 1]$, $\Delta_j \in [0, 1]$, and $k \in \mathbb{N}_+$ such that $\mu_{h(\alpha_1)} = \mu_{h(\alpha_2)}$. Then, under the clean event $\mathcal{E}$, the algorithm stops recursively invoking itself and terminates.*

Proof. We split the proof into two different cases.

1. If $|I| \leq 1/T$, then the condition at Line 3 is satisfied and thus the Algorithm 7.2 returns the pair (I, j, k), independently of the distributions at the extremes.

2. If $|I| > 1/T$, under the event $\mathcal{E}$, we have:

$$\widehat{\mu}(\alpha_2) - \widehat{\mu}(\alpha_1) \leq \mu_{h(\alpha_2)} - \mu_{h(\alpha_1)} + \frac{\Delta_j}{2} = \frac{\Delta_j}{2}.$$

Thus, the condition at Lines 13 is not satisfied and therefore the algorithm returns the tuple (I, j, k), concluding the proof. □

7.3.3 The `Optimistic-Shrink` procedure

We now introduce the second key procedure employed by Algorithm 7.1, which is `Optimistic-Shrink`, whose pseudocode is provided in Algorithm 7.3. At a high level, the algorithm takes as input an interval $I \subseteq [0, 1]$ and returns a subinterval $I' \subseteq I$ such that, for every action $\alpha \in I'$ selected during the following epoch $j + 1$, it holds that $u(\alpha) \geq \text{OPT} - \mathcal{O}(\Delta_j)$. To do so, Algorithm 7.3 relies on the estimates of the expected values at the extremes of the interval I, computed by Algorithm 7.2. As a result, Algorithm 7.3 does *not* require the learner to interact with environment, as it just manipulates the intervals in $\mathcal{I}_j$. Thus, the regret suffered by the algorithm is zero.

Depending on the length of the interval in input, Algorithm 7.3 works in two different ways.

1. If the length of the input interval $I = [\alpha_1, \alpha_2]$ is such that $\alpha_2 - \alpha_1 > 1/T$, then Algorithm 7.3 returns a subinterval $I' \subseteq I$ such that, if $\alpha \in I'$, then the value of $\ell(\alpha)\widehat{\mu}(\alpha_1) + 2\Delta_j + 2/T$ is larger than $\text{OPT}(j)$ (see Line 5). This is because, thanks to Lemma 7.1 and the definition of the clean event $\mathcal{E}$, the value $\ell(\alpha)\widehat{\mu}(\alpha_1) + 2\Delta_j + 2/T$ represents an optimistic estimate of the expected utility in each $\alpha \in I$. Therefore, if such an optimistic estimate is smaller than $\text{OPT}(j)$, then the learner's expected utility will be suboptimal at that given α, as a consequence of the relationship between $\text{OPT}(j)$ and OPT, which we show in the following Lemma 7.4.

2. If the length of the input interval $I = [\alpha_1, \alpha_2]$ is such that $\alpha_2 - \alpha_1 \leq 1/T$, then Algorithm 7.3 returns the interval itself only if the optimistic estimate of the learner's expected utility at the right extreme α_2 is larger than the current estimate of the optimum $\text{OPT}(j)$ (see Line 1). This is because such an optimistic estimate, due to the monotonicity of the jumps and the fact that the input interval is sufficiently small, represents an upper bound for the learner's expected utility over the entire interval. Hence, if the estimate is smaller than $\text{OPT}(j)$, then the learner's expected utility within that interval is suboptimal, given the relationship between $\text{OPT}(j)$ and OPT in Lemma 7.4. As a result, Algorithm 7.3 does *not* return the interval I.

Next, we prove some crucial properties that Algorithm 7.3 satisfies. The first property ensures that Algorithm 7.3 always returns an interval $I \subseteq [0, 1]$ containing an optimal action $\alpha^\star$.

Algorithm 7.3 `Optimistic-Shrink`

Require: $(I = [\alpha_1, \alpha_2], \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$, $\Delta_j \in [0,1]$, and $\mathrm{OPT}(j) \in [0,1]$
1: **if** $\alpha_2 - \alpha_1 \leq \frac{1}{T} \wedge \ell(\alpha_2) \cdot \widehat{\mu}(\alpha_2) + {}^{\Delta_j}/_2 + {}^1/_T \geq \mathrm{OPT}(j)$ **then**
2: **return** I
3: **end if**
4: **if** $\alpha_2 - \alpha_1 > \frac{1}{T}$ **then**
5: $I' \leftarrow \{\alpha \in I \mid \ell(\alpha) \cdot \widehat{\mu}(\alpha_1) + 2\Delta_j + {}^2/_T \geq \mathrm{OPT}(j)\}$
6: **return** I'
7: **end if**
8: **return** $\varnothing$

Formally:

Lemma 7.3. *Under the event $\mathcal{E}$, for every epoch $j \in \mathbb{N}_+$, it holds $\alpha^\star \in I$ for some $I \in \mathcal{I}_j$.*

Proof. To prove the lemma we employ an inductive argument. When $j = 1$, the statement is trivially satisfied since $\mathcal{I}_1 = \{[0,1]\}$ and $\alpha^\star \in [0,1]$. Suppose that $\alpha^\star \in I$ for some $I \in \mathcal{I}_j$ with $j > 1$, we prove that $\alpha^\star \in I$ for some $I \in \mathcal{I}_{j+1}$. To show that, we consider two different cases.

1. If $\alpha^\star \in I$ for some $I \in \mathcal{I}_j$ with $|I| > {}^1/_T$, then it holds:

$$\begin{aligned}
\ell(\alpha^\star)\widehat{\mu}(\alpha_1) &\geq \ell(\alpha^\star)\mu_{h(\alpha_1)} - \frac{\Delta_j}{4} \\
&\geq \ell(\alpha^\star)\mu_{h(\alpha^\star)} - \frac{7\Delta_j}{4} \\
&= \mathrm{OPT} - \frac{7\Delta_j}{4} \\
&\geq \ell(\alpha_j^\star)\mu_{h(\alpha_j^\star)} - \frac{7\Delta_j}{4} \\
&\geq \ell(\alpha_j^\star)\widehat{\mu}(\alpha_j^\star) - 2\Delta_j \\
&= \mathrm{OPT}(j) - 2\Delta_j.
\end{aligned}$$

Where the first inequality above holds under the clean event $\mathcal{E}$. The second inequality holds thanks to Lemma 7.1 since $|I| \geq {}^1/_T$, while the third inequality holds because of the optimality of OPT. The fourth inequality holds under the event $\mathcal{E}$, and the last inequality holds because of the definition of $\alpha_j^\star \in [0,1]$.

Thanks to the above inequalities, we have the following:

$$\ell(\alpha^\star)\widehat{\mu}(\alpha_1) + 2\Delta_j \geq \mathrm{OPT}(j).$$

Consequently, the condition at Line 5 is satisfied, and thus $\alpha^\star$ also belongs to some $I \in \mathcal{I}_{j+1}$.

2. If $\alpha^\star \in I$ for some $I \in \mathcal{I}_j$ with $|I| \leq 1/T$, then the following holds:

$$\begin{aligned}
\ell(\alpha_2)\widehat{\mu}(\alpha_2) &\geq \ell(\alpha_2)\mu_{h(\alpha_2)} - \frac{\Delta_j}{4} \\
&\geq \ell(\alpha_2)\mu_{h(\alpha^\star)} - \frac{\Delta_j}{4} \\
&= (\ell(\alpha_2) \pm \ell(\alpha^\star))\mu_{h(\alpha^\star)} - \frac{\Delta_j}{4} \\
&\geq \ell(\alpha^\star)\mu_{h(\alpha^\star)} - \frac{\Delta_j}{4} - \frac{1}{T} \\
&= \mathrm{OPT} - \frac{\Delta_j}{4} - \frac{1}{T} \\
&\geq \ell(\alpha_j^\star)\mu_{h(\alpha_j^\star)} - \frac{\Delta_j}{4} - \frac{1}{T} \\
&\geq \ell(\alpha_j^\star)\widehat{\mu}(\alpha_j^\star) - \frac{\Delta_j}{2} - \frac{1}{T} \\
&= \mathrm{OPT}(j) - \frac{\Delta_j}{2} - \frac{1}{T}
\end{aligned}$$

We notice that the first and last inequalities above hold under the clean event $\mathcal{E}$. The second inequality holds since $\mu_{h(\alpha_2)} \geq \mu_{h(\alpha^\star)}$ according to monotonicity. The third inequality holds by hypothesis observing that $|\alpha^\star - \alpha_1| \leq |I| \leq 1/T$. As a result, the following holds:

$$\ell(\alpha_2)\widehat{\mu}(\alpha_2) + \Delta_j/2 + 1/T \geq \mathrm{OPT}(j).$$

Consequently, the condition at Lines 1 is satisfied, and thus $\alpha^\star$ also belongs to some $I \in \mathcal{I}_{j+1}$.

Finally, since both the base case holds and the inductive step is valid, we conclude the proof. □

In order to prove the lemma, we consider two different cases. The first one is when $\alpha^\star \in I$ for some $I = [\alpha_1, \alpha_2] \in \mathcal{I}_j$ with $\alpha_2 - \alpha_1 < 1/T$. Then, under the clean event $\mathcal{E}$ and by the monotonicity of jumps, it is possible to show that the optimistic estimate of the learner's expected utility at the right extreme α_2 of the interval satisfies the condition in Line 1 of Algorithm 7.3. Thus, the interval that contains $\alpha^\star$ is guaranteed to also belong to $\mathcal{I}_{j+1}$. The second case is when $\alpha^\star \in I$ for some $I \in \mathcal{I}_j$ whose length is larger than $1/T$. Then, by employing Lemma 7.1, it is possible to show that the optimistic estimate of the learner's expected utility for action $\alpha^\star$ (evaluated in Line 5 of Algorithm 7.3) is greater than or equal to $\mathrm{OPT}(j)$. Thus, if there exists an interval $I \in \mathcal{I}_j$ such that $\alpha^\star \in I$, then there is also some $I' \in \mathcal{I}_{j+1}$ such that $\alpha^\star \in I'$. Finally, by using an inductive argument, the statement of the lemma is immediately proved.

Thanks to Lemma 7.3, it is possible to show that the difference between the optimal value OPT and the one estimated at epoch $j \in \mathbb{N}_+$, namely $\mathrm{OPT}(j)$, is proportional to Δ_j. Formally, we have:

Lemma 7.4. *Under the clean event $\mathcal{E}$, it holds*

$$\mathrm{OPT}(j) \geq \mathrm{OPT} - \tfrac{7\Delta_j}{4} - \tfrac{1}{T}$$

for every epoch $j \in \mathbb{N}_+$.

Proof. We split the proof into two cases.

1. If $\alpha^\star \in I$ with $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$ and $|I| > 1/T$, then we have:

$$\begin{aligned}
\mathrm{OPT}(j) &\geq \ell(\alpha_1)\widehat{\mu}(\alpha_1) \\
&\geq \ell(\alpha_1)\mu_{h(\alpha_1)} - \frac{\Delta_j}{4} \\
&\geq \ell(\alpha_1)\mu_{h(\alpha^\star)} - \frac{7\Delta_j}{4} \\
&\geq \ell(\alpha^\star)\mu_{h(\alpha^\star)} - \frac{7\Delta_j}{4} \\
&= \mathrm{OPT} - \frac{7\Delta_j}{4}.
\end{aligned}$$

   The first inequality above holds according to the definition of $\mathrm{OPT}(j)$, the second inequality holds under the event $\mathcal{E}$, the third inequality holds thanks to Lemma 7.1 and the last inequality holds since $\alpha^\star \geq \alpha_1$.

2. If $\alpha^\star \in I$ with $(I, \widehat{\mu}(\alpha_1), \widehat{\mu}(\alpha_2)) \in \mathcal{T}_j$ and $|I| \leq 1/T$, under the event $\mathcal{E}$, we have:

$$\begin{aligned}
\mathrm{OPT}(j) &\geq \ell(\alpha_2)\widehat{\mu}(\alpha_2) \\
&\geq \ell(\alpha_2)\mu_{h(\alpha_2)} - \frac{\Delta_j}{4} \\
&= (\ell(\alpha_2) \pm \ell(\alpha^\star))\mu_{h(\alpha^\star)} - \frac{\Delta_j}{4} \\
&\geq \ell(\alpha^\star)\mu_{h(\alpha^\star)} - \frac{\Delta_j}{4} - \frac{1}{T} \\
&= \mathrm{OPT} - \frac{\Delta_j}{4} - \frac{1}{T}.
\end{aligned}$$

   The first inequality holds according to the definition of $\mathrm{OPT}(j)$, the second inequality holds under the event $\mathcal{E}$, the third inequality holds thanks to monotonicity and the last inequality holds since $|I| \leq 1/T$.

Thanks to the above inequalities the lemma holds. □

In order to prove Lemma 7.4, we consider two different cases, which are analogous to those in the proof of Lemma 7.3. When $\alpha_2 - \alpha_1 < 1/T$, it is possible to prove the statement of the lemma by employing the definition of $\mathrm{OPT}(j)$ and Lemma 7.1. In contrast, when $\alpha_2 - \alpha_1 \geq 1/T$, the statement follows from the monotonicity of the jumps and the definition of $\mathrm{OPT}(j)$. Finally, we introduce the main property satisfied by Algorithm 7.3, which establishes a lower bound on the learner's expected utility for the intervals returned by the algorithm.

Formally:

Lemma 7.5. *Under the event $\mathcal{E}$, for every epoch $j \in \mathbb{N}_+$ and interval $I = [\alpha_1, \alpha_2] \in \mathcal{I}_{j+1}$, it holds:*

1. *If $\alpha_2 - \alpha_1 > 1/T$, then $u(\alpha) \geq \text{OPT} - 4\Delta_j - 2/T$ for every action $\alpha \in I$.*
2. *If $\alpha_2 - \alpha_1 \leq 1/T$, then $u(\alpha_2) \geq \text{OPT} - 4\Delta_j - 2/T$.*

Proof. We split the proof into two cases.

1. If $|I| > 1/T$, then under the event $\mathcal{E}$, the following holds:

$$\begin{aligned}
\ell(\alpha)\mu_{h(\alpha)} &\geq \ell(\alpha)\mu_{h(\alpha_1)} \\
&\geq \ell(\alpha)\widehat{\mu}(\alpha_1) - \frac{\Delta_j}{4} \\
&\geq \text{OPT}(j) - \frac{\Delta_j}{4} - 2\Delta_j - \frac{2}{T} \\
&\geq \text{OPT} - 4\Delta_j - \frac{2}{T}.
\end{aligned}$$

   We notice that the first inequality above holds as a consequence of monotonicity, and the second inequality holds under the clean event $\mathcal{E}$. Furthermore, if $\alpha \in I$ for some $I \in \mathcal{I}_{j+1}$, then the condition at Lines 5 is satisfied, and thus the third inequality above holds. Finally, the last inequality holds as a consequence of Lemma 7.4.

2. If $|I| \leq 1/T$, we have:

$$\begin{aligned}
\ell(\alpha_2)\mu_{h(\alpha_2)} &\geq \ell(\alpha_2)\widehat{\mu}(\alpha_2) - \frac{\Delta_j}{4} \\
&\geq \text{OPT}(j) - \frac{\Delta_j}{4} - \frac{\Delta_j}{2} - \frac{1}{T} \\
&\geq \text{OPT} - 4\Delta_j - \frac{2}{T}.
\end{aligned}$$

   The first inequality above holds under the clean event $\mathcal{E}$. The second inequality holds true because the condition at Lines 1 is satisfied.

   Finally, the last inequality holds as a consequence of Lemma 7.4, concluding the proof. □

Notice that, thanks to Lemma 7.5, every action $\alpha \in \mathcal{I}_{j+1}$ chosen by the algorithm at epoch $j+1$ is $\mathcal{O}(\Delta_j)$-optimal. Thus, since the `Find-Jumps` procedure employs a number of samples proportional to $1/\Delta_{j+1}^2$, the resulting regret suffered by the algorithm at epoch $j+1$ is proportional to $\mathcal{O}\left(\Delta_j/\Delta_{j+1}^2\right) = \mathcal{O}\left(1/\Delta_{j+1}\right)$, since $2\Delta_{j+1} = \Delta_j$.

7.3.4 Putting all together

In order to derive the regret guarantees of Algorithm 7.1, we first need to bound the number of calls made to `Find-Jumps` (including recursive ones) during each

epoch $j \in \mathbb{N}_+$, since each call may require the learner to select two actions for a number of rounds proportional to $1/\Delta_j^2$. Formally:

Lemma 7.6. *Under the clean event $\mathcal{E}$, at each epoch $j \in \mathbb{N}_+$ of Algorithm 7.1, the number of calls to `Find-Jumps` is upper bounded by $(j+2)n\lceil \log_2(T) \rceil$.*

Proof. In the following, we introduce $\mathcal{N}(j,k)$, which corresponds to the set of tuples given as input to Algorithm 7.2 at iteration $j \in \mathbb{N}_+$ and level $k \in \mathbb{N}_+$. Formally, given $j, k \in \mathbb{N}_+$, we define

$$\mathcal{N}(j,k) \coloneqq |\{I \subseteq [0,1] \mid (I,j,k) \text{ is given as input to Algorithm 7.2}\}|$$

Furthermore, we denote $N(j,k) \coloneqq |\mathcal{N}(j,k)|$. Intuitively, the quantity $N(j,k)$ corresponds to the number of nodes at level $k \in \mathbb{N}_+$ in the recursive tree generated by the execution of Algorithm 7.2. (See Figure 7.3). As a result, the quantity $\sum_{k \in \mathbb{N}_+} N(j,k)$ represents the number of calls performed to Algorithm 7.2 during the execution of the for-loop at Lines 2 in Algorithm 7.1 at iteration $j \in \mathbb{N}_+$.

We also observe that, when Algorithm 7.2 receives a tuple as input, it either recursively invokes itself twice or it does not recursively invoke itself anymore. Thus, each node in the recursive tree in Figure 7.3 has either zero or two children. This means that the quantity $N(j,k+1)/2$ represents the number of times the algorithm has recursively invoked itself at level $k \in \mathbb{N}_+$ and iteration $j \in \mathbb{N}_+$. Consequently, it represents the number of nodes that have two children at level $k \in \mathbb{N}_+$ in the tree of Figure 7.3.

We also introduce the set $\mathcal{M}(j,k)$, which contains all the tuples corresponding to the leaves of the recursive tree generated by the execution of Algorithm 7.2 (see the red boxes in Figure 7.3) at level $k \in \mathbb{N}_+$ and iteration $j \in \mathbb{N}_+$. Formally, we have:

$$\mathcal{M}(j,k) \coloneqq \mathcal{N}(j,k) \cup \{I \subseteq [0,1], k' \in [k-1] \mid (I,j,k') \text{ is an input of Algo 7.2 s.t. it does not invoke itself}\}.$$

We denote $M(j,k) \coloneqq |\mathcal{M}(j,k)|$ and we observe that can be recursively expressed as follows:

$$M(j,k+1) = \underbrace{M(j,k) - \frac{N(j,k+1)}{2}}_{(1)} + 2\underbrace{\frac{N(j,k+1)}{2}}_{(2)} = M(j,k) + \frac{N(j,k+1)}{2}, \tag{7.1}$$

for all $k \geq 1$, with $M(j,1) = |\mathcal{I}_j|$. As observed before, the quantity (2) in Equation 7.1 represents the number of nodes at level k that have two children and thus will be leaf nodes at level $k+1$. The quantity in (1) in Equation 7.1 represents all the leaf nodes at level k that have no children and thus will remain leaves at level $k+1$ as well.

Furthermore, for each $j \in \mathbb{N}_+$, the following holds.

1. If $k = 1$, then $N(j, 1) = |\mathcal{I}_j|$. This is because Algorithm 7.2 receives as input all the tuples $(I, j, 1)$ with $I \in \mathcal{I}_j$, as specified at Lines 2 in Algorithm 7.1.

2. If $k > \lceil \log_2(T) \rceil + 1$, then $N(j, k) = 0$. When Algorithm 7.2 receives a tuple (I, j, k) as input and it recursively invokes itself with a new tuple $(I', j, k+1)$, the length of I' is halved compared to that of I. Thus, observing that Algorithm 7.2 does not invoke itself again if it receives an interval $I \subseteq [0, 1]$ that satisfies $|I| \leq 1/T$, we conclude that there is no interval I such that the tuple $(I, \lceil \log_2(T) \rceil + 2, j)$ may represent a valid input to Algorithm 7.2. Indeed, any interval $I \subseteq [0, 1]$, when halved $\lceil \log_2(T) \rceil$ times, will ultimately have a length smaller than $1/T$.

3. If $0 < k \leq \lceil \log_2(T) \rceil + 1$, then $N(j, k) \leq 2n$. This is because when Algorithm 7.2 receives an interval $I = [\alpha_1, \alpha_2]$ as input, it recursively invokes itself only if the condition $\mu_{h(\alpha_1)} < \mu_{h(\alpha_2)}$ holds, as stated in Lemma 7.2, and thus only if there is at least one jump discontinuity within the interval. Furthermore, observing that there are at most $n - 1$ jumps and that Algorithm 7.2 recursively invokes itself twice, we have $N(j, k) \leq 2(n-1) \leq 2n$.

Then, thanks to Equation 7.1 and the above observations, we have that:

$$M(j,k) = \begin{cases} M(j,k+1) \leq M(j,k) \quad \forall\, k > \lceil \log_2(T) \rceil, \\ M(j,k+1) \leq M(j,k) + n \quad \forall\, 1 \leq k \leq \lceil \log_2(T) \rceil, \\ M(j,1) = |\mathcal{I}_j|. \end{cases}$$

Consequently, at each iteration $j \in \mathbb{N}_+$, we have:

$$M(j,k) \leq n\lceil \log_2(T) \rceil + |\mathcal{I}_j|,$$

for all $k > 0$.

We also observe that $|\mathcal{T}_j| = M(j, \lceil \log_2(T) \rceil + 1)$. This is because, the set $\mathcal{T}_j$ contains all the leaf nodes of the recursive tree generated by the execution of Algorithm 7.2. (See Line 2 in Algorithm 7.1). In addition, Algorithm 7.3, can only decrease the number of intervals belonging to $\mathcal{I}_{j+1}$ compared to the number of tuples belonging to $\mathcal{T}_j$. Thus, thanks to the latter observations, we have that:

$$|\mathcal{I}_{j+1}| \leq |\mathcal{T}_j| \leq n\lceil \log_2(T) \rceil + |\mathcal{I}_j|$$

and $|\mathcal{I}_1| = 1$. Then, by induction, we can easily show that:

$$|\mathcal{I}_j| \leq (j-1)n\lceil \log_2(T) \rceil + 1 \leq jn\lceil \log_2(T) \rceil,$$

for each $j \in \mathbb{N}$. We prove that, for each $j \geq 1$, the number of calls to Algorithm 7.2 satisfies:

$$\sum_{k \in \mathbb{N}_+} N(j,k) = N(j,1) + \sum_{k=2}^{\lceil \log_2(T) \rceil + 1} N(j,k) \leq |\mathcal{I}_j| + 2n\lceil \log_2(T) \rceil \leq (j+2)n\lceil \log_2(T) \rceil,$$

concluding the proof. □

To prove Lemma 7.6, we observe that in each epoch $j \in \mathbb{N}_+$, thanks to Lemma 7.2, `Find-Jumps` recursively invokes itself only if the distributions at the extremes of the input interval are different and, thus, if the input interval contains at least one jump discontinuity. Since the number of discontinuities is bounded by $n-1$, at each level $k \in \mathbb{N}$ of the tree of recursive calls generated by the execution of `Find-Jumps` in epoch $j \in \mathbb{N}_+$ (see Figure 7.3), the procedure recursively invokes itself invoked at most $2(n-1)$ times. Furthermore, given the termination condition at Lines 7 in Algorithm 7.2, if the size of the input interval is smaller than $1/T$, then the procedure does *not* invoke itself anymore. Thus, by observing that `Find-Jumps` recursively invokes itself with an input interval whose length is halved, we have that the maximum depth $k \in \mathbb{N}_+$ reached in the tree of recursive calls at each epoch is at most $\log_2(T)$. Therefore, by combining the above observations, the number of times that `Find-Jumps` is invoked during epoch $j \in \mathbb{N}_+$ is bounded by $2n\log_2(T) + |\mathcal{I}_j|$, as the procedure is initially invoked on each interval in $\mathcal{I}_j$ by Algorithm 7.1. Finally, by employing an inductive argument, it is possible to show that $|\mathcal{I}_j| \leq (j+2)n\log_2(T)$, thus providing an upper bound on the number of calls to `Find-Jumps` in each epoch $j \in \mathbb{N}_+$.

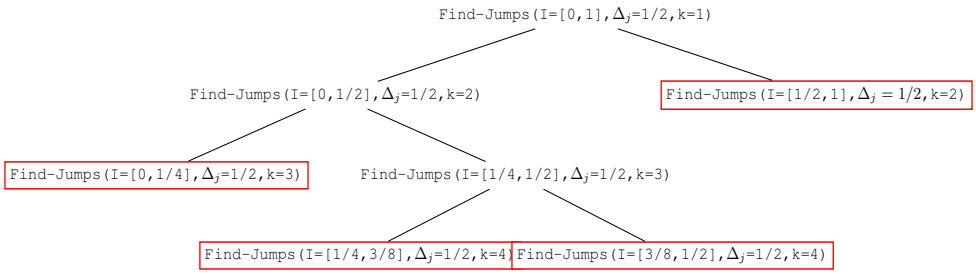

Figure 7.3: *Example of tree of recursive calls generated by the execution of* `Find-Jumps` *(Algorithm 7.2).*

We are now ready to prove the main theorem of this chapter.

Theorem 7.1. *The regret suffered by Algorithm 7.1 is bounded by* $\mathcal{O}(\sqrt{nT}\log^2(T))$.

Proof. Let $\eta > 0$. We observe that at iteration $j \in \mathbb{N}_+$, with probability $1-\eta$, by applying Hoeffding's inequality and a union bound, Algorithm 7.2 computes two estimates satisfying:

$$|\widehat{\mu}(\alpha_i) - \mu_{h(\alpha_i)}| \leq \frac{\Delta_j}{4}, \tag{7.2}$$

for each $i = 1, 2$, using $8/\Delta_j^2 \log(4/\eta)$ samples for each extreme of the interval.

Thus, employing a union bound and observing that Algorithm 7.2 is called at most T times, we have that the condition in Equation (7.2) is satisfied with probability $1-\eta T$ during the execution of Algorithm 7.1. By setting $\eta T = \delta$, we ensure that with $8/\Delta_j^2 \log(4T/\delta)$ samples from both extremes, the condition in Equation (7.2) is satisfied with probability $1-\delta$ whenever Algorithm 7.2 is

invoked. Thus, $\mathbb{P}(\mathcal{E}) \geq 1 - \delta$.

In the following, we denote with $\mathcal{P}_j$ the number of rounds required to execute iteration $j \in \mathbb{N}_+$. We notice that $\sum_{j \in \mathbb{N}_+} \mathcal{P}_j = T$, and, by Lemma 7.6, we have:

$$\mathcal{P}_j \leq \underbrace{2\frac{8}{\Delta_j^2}\log\left(4T/\delta\right)}_{(a)} \underbrace{(j+2)n\lceil\log_2(T)\rceil}_{(b)} = \frac{16}{\Delta_j^2}\log\left(4T/\delta\right)(j+2)n\lceil\log_2(T)\rceil$$

Where (a) represents an upper bound to number of samples required to execute Algorithm 7.2 at iteration $j \in \mathbb{N}_+$, while (b) is a upper bound to the number of calls to Algorithm 7.2 at iteration $j \in \mathbb{N}_+$ according to Lemma 7.6.

Furthermore, under the clean event $\mathcal{E}$, at each iteration $j \in \mathbb{N}_+$, the learner selects actions that are $(4\Delta_{j-1} + 2/T)$-optimal, as shown in Lemma 7.5. Since $\Delta_{j-1} = 2\Delta_j$, and $\Delta_j = 1/2^j$, this implies that the actions selected by the learner are $(8\Delta_j + 2/T)$-optimal. Consequently, the cumulative regret can be upper bounded as follows:

$$\begin{aligned} R_T &\leq \left(\sum_{j \in \mathbb{N}_+} \mathcal{P}_j \left(8\Delta_j + \frac{2}{T}\right)\right)\mathbb{P}(\mathcal{E}) + T \cdot \mathbb{P}(\mathcal{E}^{\mathsf{C}}) \\ &\leq \sum_{j \in \mathbb{N}_+} 8\Delta_j \mathcal{P}_j + 2 + T \cdot \delta \\ &= \underbrace{\sum_{j \leq j^\star} 8\Delta_j \mathcal{P}_j}_{(*)} + \underbrace{\sum_{j > j^\star} 8\Delta_j \mathcal{P}_j}_{(**)} + 2 + T \cdot \delta, \end{aligned} \tag{7.3}$$

where we let $j^\star = \max\left\{0, \left\lfloor \log_2\left(\sqrt{\frac{T}{n}}\frac{1}{\log_2^2(T)}\right)\right\rfloor\right\}$.

We separately bound the two terms $(*)$ and $(**)$ in Equation (7.3).

- Bound on $(*)$. Thanks to the definition of $j^\star \in \mathbb{N}_+$, we have

  $$\Delta_j \geq \log_2^2(T)\sqrt{\frac{n}{T}}$$

  for each $j \in [j^\star]$. Thus, we have:

  $$\begin{aligned} \sum_{j \leq j^\star} 8\Delta_j \mathcal{P}_j &\leq \sum_{j \leq j^\star} \frac{128}{\Delta_j}\log\left(4T/\delta\right)(j+2)n\lceil\log_2(T)\rceil \\ &\leq \frac{128\sqrt{nT}\log\left(4T/\delta\right)\lceil\log_2(T)\rceil\sum_{j \leq j^\star}(j+2)}{\log_2^2(T)} \\ &\leq \frac{128\sqrt{nT}\log\left(4T/\delta\right)\lceil\log_2(T)\rceil\left(\log_2^2(T) + 3\log_2(T)\right)}{\log_2^2(T)} \\ &\leq \mathcal{O}\left(\sqrt{nT}\log\left(4T/\delta\right)\log(T)\right). \end{aligned}$$

- Bound on $(**)$. Thanks to the definition of $j^\star \in \mathbb{N}_+$, it holds

$$\Delta_j \leq \log_2^2(T)\sqrt{\tfrac{n}{T}}$$

for each $j > j^\star$. Thus, we can prove that the following holds:

$$\sum_{j>j^\star} 8\Delta_j \mathcal{P}_j \leq 8\sqrt{\frac{n}{T}}\log_2^2(T)\sum_{j>j^\star}\mathcal{P}_j \leq 8\sqrt{nT}\log_2^2(T) \leq \mathcal{O}(\sqrt{nT}\log^2(T)).$$

Putting $(*)$ and $(**)$ together and setting $\delta = 1/T$ we have:

$$R_T \leq \mathcal{O}(\sqrt{nT}\log^2(T)),$$

concluding the proof. □

Next, we provide a sketch of how to prove that the regret suffered by Algorithm 7.1 is of the order of $\widetilde{\mathcal{O}}(\sqrt{T})$. Notice that, thanks to Lemma 7.5, the regret suffered during the execution of `Find-Jumps` in epoch $j \in \mathbb{N}_+$ is $\mathcal{O}(1/\Delta_j) = \mathcal{O}(2^j)$. Furthermore, thanks to Lemma 7.6, we have that `Find-Jumps` is invoked a number of times proportional to the index of the epoch $j \in \mathbb{N}_+$. Therefore, the regret suffered during the execution of `Find-Jumps` in epoch $j \in \mathbb{N}_+$ is bounded by $\mathcal{O}(j2^j)$. Let us also observe that the largest epoch $j^\star \in \mathbb{N}_+$ executed by Algorithm 7.1 must be such that $j^\star \leq \log_2(T)/2$. Otherwise, the `Find-Jumps` procedure would require a number of rounds larger than $1/\Delta_{j^\star}^2 = T$ to be executed. Therefore, the regret suffered by `Find-Jumps` can be bounded by:

$$\sum_{j \leq \log_2(T)/2} j2^j \leq \widetilde{\mathcal{O}}(\sqrt{T}).$$

In conclusion, thanks to Theorem 7.1 and the relations between BwMJ problems and microeconomic models discussed in Section 7.2, we have that the two following corollaries hold.

Corollay 7.1. *In hidden-action principal-agent problems, there exists an algorithm that achieves a regret of $\widetilde{\mathcal{O}}(\sqrt{nT})$ when competing against an optimal linear contract, where n is the number of agent's actions.*

In hidden-action principal-agent problems, Zhu et al. [2023] proposed an algorithm that achieves a regret of $\widetilde{\mathcal{O}}(T^{2/3})$ when competing against an optimal linear contract. Zhu et al. [2023] leave open the question of whether such regret dependence could be improved when the number of the agent's actions is sufficiently smaller compared to the time horizon T. We provide a positive answer to this question in Corollary 7.1.

Corollay 7.2. *In posted-price auctions with finitely many buyer valuations, there exists an algorithm that achieves a regret of $\widetilde{\mathcal{O}}(\sqrt{nT})$ when competing against an optimal fixed price, where n is the number of possible buyer valuation values.*

In posted-price auctions with finitely many buyer valuations, Cesa-Bianchi et al. [2019] proposed an algorithm that achieves a regret of $\widetilde{\mathcal{O}}(\sqrt{nT} + V(V+1))$, where

V is an instance-dependent parameter encoding the magnitude of jump gaps. Cesa-Bianchi et al. [2019] leave open the question of whether the dependence on such a parameter V is avoidable or not. We positively answer this question in Corollary 7.2.

7.4 On the optimality of the RJI−OS algorithm

In this section, we show that the regret guarantees attained by the RJI−OS algorithm are optimal. Specifically, the regret attained by Algorithm 7.1 is optimal even when restricting the attention to BwMJ instances related to microeconomic settings, namely those capturing hidden-action principal-agent problems and posted-price auctions, which have been introduced in Section 7.2.

Specifically, the RJI−OS algorithm, when employed in posted-price auctions settings (see Section 7.2.1), achieves optimal regret guarantees that match the lower bound introduced by Cesa-Bianchi et al. [2019]. Similarly, when employed in principal-agent problems (see Section 7.1.3), the RJI−OS algorithm attains optimal regret guarantees, matching a lower bound we discuss in the following.

We now present a lower bound for principal-agent settings. Specifically, for the problem of minimizing the regret with respect to an optimal-in-hindsight *linear contract* (see Section 7.1.3 for its formal introduction). Our result is formally stated in the following Theorem 7.2. Notice that Theorem 7.2 strengthens an existing lower bound by Zhu et al. [2023], which holds only when the agent has a number of actions proportional to $T^{1/3}$, whereas Theorem 7.2 holds for any number of agent's actions smaller than or equal to that needed by Zhu et al. [2023]. Formally:

Theorem 7.2. *Let $n, T \in \mathbb{N}_+$ with $T^{1/3} \geq n \geq 3$. Then, for any learner's policy $\pi := (\pi_t)_{t\in[T]}$, there exists an instance of BwMJ problem, encoding an hidden-action principal-agent problem where the agent has n actions available (see Section 7.1.3), in which the regret attained by the policy π is*

$$R_T(\pi) \geq \Omega\left(\sqrt{nT}\right).$$

Proof. Let $\epsilon \in (0,1)$ and $k \in (2,3)$ be two parameters to be defined in the following. We consider a principal-agent instance $\mathcal{P}$ characterized by two outcomes $r_1 = 1$ and $r_2 = 0$ and by $n \geq 3$ agent's actions. The costs and the corresponding probability distributions over outcomes of these actions are defined as follows.

$$\begin{cases} F_1 = (0,1) & c_1 = 0 \\ F_2 = \left(\frac{1}{2} + \frac{1}{2}\epsilon, \frac{1}{2} - \frac{1}{2}\epsilon\right) & c_2 = (1+\epsilon)\left(\frac{1}{2} - \frac{1}{k}\right) \\ F_i = \left(\frac{1}{2} + \epsilon(i-2), \frac{1}{2} - \epsilon(i-2)\right) & c_i = c_{i-1} + \alpha_i\left(F_{i,1} - F_{i-1,1}\right) \quad i \geq 3 \end{cases}$$

Where the parameters $\alpha_i \in [0,1]$ are given by:

$$\alpha_1 = 0 \quad \text{and} \quad \alpha_i := 1 - \frac{1}{\left(\frac{1}{2} + \epsilon(i-2)\right)k} \quad i = 2, \dots, n.$$

In the following, we assume $\epsilon n \leq 1/4$ to ensure that the probability distributions and the costs of the different actions are well-defined. Indeed, for each $i \geq 3$, we have:

$$c_i = c_{i-1} + \alpha_i \left(F_{i,1} - F_{i-1,1}\right) \leq c_{i-1} + \epsilon.$$

The last inequality holds observing that $|\alpha_i| \leq 1$ for each $i \in [n]$ and $|F_{i,1} - F_{i-1,1}| \leq \epsilon$ for each $i \geq 3$. Thus, thanks to the assumption that $\epsilon n \leq 1/4$, it is easy to verify that $c_n \leq 1$ by employing a simple recursive argument and noticing that $c_2 \leq 1/3$ since $\epsilon \leq 1$ and $k \leq 3$.

We also notice that each action $i \geq 2$ provides the agent with an expected utility greater or equal to what they would achieve by selecting action $i-1$ when the principal commits to a contract $\alpha \geq \alpha_i$. This is because $F_{i,1} \geq F_{i-1,1}$ for each $i \geq 2$, and, by a simple calculation, it is possible to verify that the following equalities hold:

$$F_{i-1,1}\alpha_i - c_{i-1} = F_{i,1}\alpha_i - c_i \;\text{ for each }\; i = 2, \ldots, n.$$

Thus, each action $i \geq 2$ provides the agent with a higher expected utility compared to all the actions $i' \in [n]$ with $i' < i$ when $\alpha \geq \alpha_i$. At the same time, the action $i+1$ provides the agent higher utility compared to action i only when $\alpha \geq \alpha_{i+1}$. Consequently, the agent's best-response regions are given by:

$$\mathcal{B}_i = [\alpha_i,\, \alpha_{i+1}) \;\; i = 1, \ldots, n-1 \quad \text{and} \quad \mathcal{B}_n = [\alpha_n,\, 1]\,, \tag{7.4}$$

as we assume the agent breaks ties in favor of the principal.

We also observe that for each $\alpha \in \mathcal{B}_i$ and action $i \in [n]$, the principal's expected utility satisfies $u(\alpha) \leq u(\alpha_i)$, as $u(\alpha)$ is a decreasing function in $\mathcal{B}_i$. Furthermore, the principal's expected utility in each α_i, with $i \geq 3$, is such that:

$$u(\alpha_i) = (1-\alpha_i)F_{i,1} = \frac{1}{\left(\frac{1}{2} + \epsilon(i-2)\right)k}\left(\frac{1}{2} + \epsilon(i-2)\right) = \frac{1}{k}.$$

Similarly, when $\alpha = \alpha_2$, the principal's expected utility is equal to:

$$u(\alpha_2) = (1-\alpha_2)F_{2,1} = \frac{2}{k}\left(\frac{1+\epsilon}{2}\right) = \frac{1+\epsilon}{k}.$$

Finally, we notice that $u(\alpha_1) = 0$. Thus, in instance $\mathcal{P}$, the optimal (linear) contract is equal to α_2.

We denote with $\mathbb{P}$ the probability distribution over all possible histories associated with the execution of π in instance $\mathcal{P}$ over $T \geq 0$ rounds. Similarly, we define $\mathbb{E}$ as the expectation under the probability distribution $\mathbb{P}$. We also denote with $T_i(T)$ the number of rounds in which the agent selects action $i \in [n]$ out of $T \geq 0$ rounds, and we let:

$$i^\star = \arg\min_{i>2} \mathbb{E}\left[\, T_i(T) \,\right]. \tag{7.5}$$

Intuitively, the action $i^\star > 2$ corresponds to the (sub-optimal) action that is selected less frequently by the agent when the policy π is executed in instance $\mathcal{P}$ over $T \geq 0$ rounds.

We now introduce a different principal-agent instance $\mathcal{P}'$ depending on the action $i^\star$ defined in Equation (7.5). This new instance shares the same distribution over the set of outcomes as instance $\mathcal{P}$, with the main difference being that, in instance $\mathcal{P}'$, the probability of observing the first outcome when the agent selects action $i^\star$ is increased by $\epsilon > 0$. Formally, we have:

$$\begin{cases} F'_1 = (0, 1) & c'_1 = 0 \\ F'_2 = \left(\frac{1}{2} + \frac{1}{2}\epsilon, \frac{1}{2} - \frac{1}{2}\epsilon\right) & c'_2 = (1+\epsilon)\left(\frac{1}{2} - \frac{1}{k}\right) \\ F'_{i^\star} = \left(\frac{1}{2} + \epsilon(i^\star - 1), \frac{1}{2} - \epsilon(i^\star - 1)\right) & c'_{i^\star} = c'_{i^\star - 1} + \alpha_{i^\star}\left(F'_{i^\star,1} - F'_{i^\star - 1,1}\right) \\ F'_i = \left(\frac{1}{2} + \epsilon(i-2), \frac{1}{2} - \epsilon(i-2)\right) & c'_i = c'_{i-1} + \alpha_i\left(F'_{i,1} - F'_{i-1,1}\right) \quad i \geq 3. \end{cases}$$

We observe that, in instance $\mathcal{P}'$, the actions $i^\star$ and $i^\star + 1$ coincide when $i^\star \neq n$. Therefore, it is without loss of generality to assume that when the principal selects a contract $\alpha \in \mathcal{B}_{i^\star}$, the agent chooses action $i^\star$, while when the principal selects $\alpha \in \mathcal{B}_{i^\star+1}$, the agent chooses action $i^\star + 1$. Using the same argument employed above, it is possible to verify that the agent's best-response regions $\mathcal{B}'_i$ in instance $\mathcal{P}'$ satisfy:

$$\mathcal{B}'_i = \mathcal{B}_i \quad \forall i \in [n].$$

Moreover, we notice that the principal's utility in instance $\mathcal{P}'$, evaluated at $\alpha_{i^\star}$, is equal to:

$$\begin{aligned} u'(\alpha_{i^\star}) = (1 - \alpha_{i^\star})F_{i^\star,1} &= \frac{1}{\left(\frac{1}{2} + \epsilon(i^\star - 2)\right)k}\left(\frac{1}{2} + \epsilon(i^\star - 2) + \epsilon\right) \\ &= \frac{1}{k}\left(1 + \frac{\epsilon}{\left(\frac{1}{2} + \epsilon(i^\star - 2)\right)}\right) \\ &\geq \frac{1}{k}\left(1 + \frac{4}{3}\epsilon\right), \end{aligned}$$

where the last inequality holds since $\epsilon \leq \epsilon n \leq 1/4$. Furthermore, the principal's utility evaluated at the remaining α_i is equal to:

$$u'(\alpha_i) = u(\alpha_i),$$

for each $i \neq i^\star$. Thus, in instance $\mathcal{P}'$, the optimal contract coincides with $\alpha_{i^\star}$. In the following, we assume that whenever the principal selects a contract $\alpha \in \mathcal{B}_i$ (or $\alpha \in \mathcal{B}'_i$), they achieve an expected utility equal to what they achieve by selecting α_i. This assumption only facilitates the learning problem faced by the principal, since the principal's expected utility is a decreasing function in each agent's best-response region. Moreover, the feedback received by the principal is equivalent for each action $\alpha \in \mathcal{B}_i$ (or $\alpha \in \mathcal{B}'_i$).

In this way, the learning problem faced by the principal reduces to a multi-armed bandit problem, where the set of arms available in the two instances coincides with the set $\{\alpha_i\}_{i=1}^n$, and each arm α_i provides utility $u(\alpha_i)$ and $u'(\alpha_i)$ in instances $\mathcal{P}$ and $\mathcal{P}'$, respectively. Thus, the regret suffered in instance $\mathcal{P}$ can be lower bounded as follows:

$$R_T(\pi) \geq \mathbb{P}\left(T_2(T) \leq T/2\right) \frac{T}{2} \frac{\epsilon}{k},$$

since the optimal arm in instance $\mathcal{P}$ provides an expected utility to the principal that is ϵ/k better compared to all the contracts α_i with $i \neq 2$. Furthermore, the regret in instance $\mathcal{P}'$ can be lower-bounded as follows.

$$R'_T(\pi) \geq \mathbb{P}'\left(T_2(T) > T/2\right) \frac{T}{2} \frac{\epsilon}{3k}, \tag{7.6}$$

where, similar to instance $\mathcal{P}$, we let $\mathbb{P}'$ be the probability distribution over all possible histories when executing π in instance $\mathcal{P}'$ over $T \geq 0$ rounds and we define $\mathbb{E}'$ the expectation under $\mathbb{P}'$.

We observe that Equation (7.6) holds because α_2 provides an expected utility that is $\epsilon/3k$ worse compared to the utility achieved in $\alpha_{i^\star}$. We also observe that the average number of times the agent selects action $i^\star$ when the principal executes policy π in instance $\mathcal{P}$ can be upper bounded as follows:

$$\mathbb{E}[T_{i^\star}(T)] \leq \frac{T}{n-2}.$$

Since $\sum_{i \geq 3} \mathbb{E}[T_i(T)] \leq T$ and because of the definition of $i^\star$ introduced in Equation (7.5). Furthermore, by employing the Bretagnolle–Huber inequality, we have that:

$$\begin{aligned} R_T(\pi) + R'_T(\pi) &\geq \left(\mathbb{P}\left(T_2(T) \leq T/2\right) + \mathbb{P}'\left(T_2(T) > T/2\right)\right) \frac{T\epsilon}{6k} \\ &\geq \frac{T\epsilon}{12k} e^{-\mathcal{KL}(\mathbb{P}, \mathbb{P}')}. \end{aligned}$$

Then, by employing the divergence decomposition lemma (see **?**) and observing that the principal receives different feedback in the two instances only when they select contracts $\alpha \in \mathcal{B}_{i^\star}$, we have:

$$\begin{aligned} \mathcal{KL}\left(\mathbb{P}, \mathbb{P}'\right) &= \mathbb{E}[T_{i^\star}(T)] \cdot \mathcal{KL}\left(F_{i^\star}, F'_{i^\star}\right) \\ &\leq \frac{T}{n-2} \frac{\epsilon^2}{\left(\frac{1}{2} - \epsilon(i^\star + 1)\right)\left(\frac{1}{2} + \epsilon(i^\star - 1)\right)} \\ &\leq \frac{16}{3} \frac{T}{n-2} \epsilon^2, \end{aligned}$$

where we employed the fact that the Kullback-Leibler divergence between two Bernoulli distributions can be upper bounded as

$$\mathcal{KL}(\mathcal{B}(p), \mathcal{B}(q)) \leq \frac{(p-q)^2}{(1-q)q}$$

and the fact that $\epsilon(i^\star - 1) \leq \epsilon n \leq 1/4$. Thus, we have:

$$\begin{aligned}
\max\{R_T(\pi), R'_T(\pi)\} &\geq \frac{1}{2}\left(R_T(\pi) + R'_T(\pi)\right) \\
&\geq \frac{T\epsilon}{24k} e^{-\mathcal{KL}(\mathbb{P},\mathbb{P}')} \\
&\geq \frac{T\epsilon}{24k} \exp\left(-\frac{16T\epsilon^2}{3(n-2)}\right) \\
&\geq \frac{5e^{-1}}{1008}\sqrt{nT}
\end{aligned}$$

Where the last step holds by setting $\epsilon = \sqrt{n/16T}$, $k = 21/10$ and observing that $n/(n-2) \leq 3$ since $n \geq 3$. Finally, we have that the number of actions n must satisfy:

$$\epsilon n = \sqrt{\frac{n}{16T}} n \leq \frac{1}{4},$$

which implies that $n \leq T^{1/3}$, concluding the proof. □

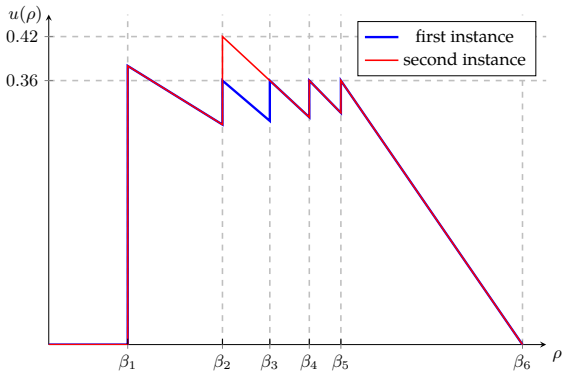

Figure 7.4: *Principal's expected utility in the two instances used to prove the lower bound in Theorem 7.2.*

In order to prove Theorem 7.2, we draw a suitable connection between regret-minimizing linear contracts in principal-agent problems and regret minimization in classical multi-armed bandits. To do so, we construct two principal-agent instances in which the distributions over outcomes of agent's actions coincide, except for one. Thus, the principal receives the same feedback and utility, except when the agent selects the action that is different in the two instances. This is made possible by suitably choosing the costs of agent's actions in the two cases. Figure 7.4 pro-

vides an example of the principal's expected utility in the two instances. We notice that, in order to distinguish between the two instances, the principal is required to play in the interval $\mathcal{B}_3 = [\beta_2, \beta_3)$, since this corresponds to the set of linear contracts under which the action different in the two instances is a best response for the agent. However, this results in a large regret in the first instance, since an optimal contract in the first instance is $\beta_1 \in [0, 1]$. Similarly, if the principal only plays a few samples from the interval $\mathcal{B}_3 = [\beta_2, \beta_3)$, then the regret in the second instance is large, since an optimal contract in such an instance is $\beta_2 \in [0, 1]$.

Let us observe that the lower bound presented in Theorem 7.2 cannot be extended to settings with a larger number of actions, except for absolute constants and logarithmic factors in the number of rounds T. Indeed, if in Theorem 7.2 the number of an agent's actions n exceeds $T^{1/3}$, then the lower bound on the regret would be $\Omega(T^{\alpha})$ for some $\alpha > 2/3$. However, this contradicts the $\widetilde{\mathcal{O}}(T^{2/3})$ upper bound on the regret achievable by employing the algorithm by Kleinberg [2004].

7.5 Related works

Ho et al. [2014] first introduce the problem of learning approximately-optimal contracts in a setting where the principal has no prior knowledge of the agent and repeatedly interacts with them. They propose an algorithm with $\widetilde{\mathcal{O}}(\sqrt{m}\, T^{2/3})$ regret with respect to an optimal linear contract, where m is the number of outcomes. Subsequently, Zhu et al. [2023] improve the result in Ho et al. [2014] by providing an algorithm that achieves an $\widetilde{\mathcal{O}}(T^{2/3})$ regret bound. They also show that this bound is tight when the agent has a number of actions n satisfying $n \propto T^{1/3}$. Duetting et al. [2023b] show that it is possible to achieve $\mathcal{O}(\log \log T)$ regret when the principal receives stronger feedback than that considered by Zhu et al. [2023]. Specifically, Duetting et al. [2023b] assume that the principal observes their expected utility in each contract they propose to the agent.

This chapter also shares key features with continuum-armed bandit problem Kleinberg [2004], Bubeck et al. [2008], Kleinberg et al. [2008] in which the learner's expected utility is a one-sided Lipschitz function. Duetting et al. [2023b] consider a regret minimization problem when the learner's function is an unknown *one-sided Lipschitz continuous function*. However, unlike our paper, they receive deterministic feedback about the principal's utility after pulling an arm. Mao et al. [2018] study the problem of learning a Lipschitz function from binary feedback applied to contextual dynamic pricing. However, different from our paper, they consider an adversarial setting. Mehta et al. [2023] introduce a threshold linear bandit problem in which, as in our problem, the learner's utility is piecewise linear. In their model, the learning action space coincides with $[0, 1]^d$ (with $d \in \mathbb{N}$) and is divided into two regions by a separating hyperplane. In contrast, our work addresses scenarios with potentially many discontinuities, when the learner's action space is limited to the one-dimensional interval $[0, 1]$.

CHAPTER 8

Learning robust contracts

In Chapters 6 and 7 we focused on settings where the agent is assumed to play an exact best response after observing the principal's commitment. However, in many real-world scenarios this assumption may not hold. For instance, the agent may lack the computational resources to compute an exact best response, or may misinterpret the contract proposed by the principal. Motivated by these considerations, in this chapter we introduce a robust formulation of the hidden-action principal-agent problem where the agent can select approximate best responses.

Specifically, we assume that the agent may choose actions that are up to $\delta \in (0, 1)$ suboptimal with respect to a best-response action in a given contract. Contracts designed under this assumption are said to be *robust*, as they are resistant to arbitrary suboptimal behavior of the agent. Within this framework, we first quantify the principal's utility loss due to robustness in contracts. We establish δ-dependent upper and lower bounds, and show that, unlike in general Stackelberg games (see, *e.g.*, [Gan et al., 2023a]), these bounds do not depend on the inducibility gap $\Delta > 0$ characterizing the problem instance.

We study settings in which the principal, who has no prior knowledge of the agent's features, interacts with an agent playing approximate best responses over multiple rounds, and we design no-regret algorithms with performance comparable to the state of the art by Zhu et al. [2023]. On the computational side, *i.e.*, when the principal has prior knowledge of the agent's features, we provide a polynomial-time algorithm for computing optimal contracts under approximate best responses. This result contrasts with the hardness of analogous commitment problems in Stackelberg games [Gan et al., 2023a] and Bayesian persuasion [Yang and Zhang, 2025].

8.1 Contract design and approximate best responses

In this chapter, we study a variation of the classical hidden-action principal-agent problem, where the agent plays an action that is an *approximate* best response. Given $\delta \in (0,1)$, we define the set $A^\delta(p) \subseteq A$ of agent's δ-*best responses* in a given contract $p \in \mathbb{R}^m_+$ as follows:

$$A^\delta(p) \coloneqq \left\{ a \in \mathcal{A} \mid \sum_{\omega \in \Omega} F_{a,\omega} p_\omega - c_a > \max_{a' \in \mathcal{A}} \sum_{\omega \in \Omega} F_{a',\omega} p_\omega - c_{a'} - \delta \right\}.$$

We adopt an adversarial robust approach, in the sense that, whenever the principal commits to a contract $p \in \mathbb{R}^m_+$, the agent selects a δ-best response that minimizes principal's expected utility, namely an action $a^\delta(p) \in A^\delta(p)$ such that

$$a^\delta(p) \in \arg\min_{a \in A^\delta(p)} \sum_{\omega \in \Omega} F_{a,\omega}(r_\omega - p_\omega).$$

We refer to the utility of a δ-*robust* contract p as $u^{\mathsf{P}}(p, a^\delta(p))$.

Given $\delta \in (0,1)$, the principal's goal is to design an *optimal* δ-robust contract $p^\star \in \mathbb{R}^m_+$, which is formally defined as:

$$p^\star \in \arg\max_{p \in \mathbb{R}^m_+} \min_{a \in A^\delta(p)} u^{\mathsf{P}}(p,a) = \arg\max_{p \in \mathbb{R}^m_+} \Psi(p), \tag{8.1}$$

where $\Psi(p) \coloneqq \min_{a \in A^\delta(p)} u^{\mathsf{P}}(p,a)$ denotes the principal's expected utility in a δ-robust contract $p \in \mathbb{R}^m_+$. Notice that analogous "δ-robust solution concepts" have been already introduced in similar settings, namely Stackelberg games [Gan et al., 2023a] and Bayesian persuasion [Yang and Zhang, 2025]. Our δ-robust contracts are their analogous in the context of hidden-action principal-agent problems.

In the following, given $\delta \in (0,1)$, we denote the expected utility of the principal in an optimal δ-robust contract $p^\star$ as $\mathrm{OPT}(\delta) \coloneqq u^{\mathsf{P}}(p^\star, a^\delta(p^\star))$. We remark that $\mathrm{OPT}(\delta)$ is always well defined, as an optimal δ-robust contract, according to the definition in Equation (8.1), always exists. Intuitively, this is due to the strict inequality in the definition of the set $A^\delta(p)$, as observed by Gan et al. [2023a] for Stackelberg games. Thus, in order to prove the existence of an optimal δ-robust contract, it is possible to employ the same argument used to prove Proposition 1 in [Gan et al., 2023a].

8.2 The price of robustness

In this section, we characterize the maximum utility that the principal can obtain through robust contracts. Specifically, we establish upper and lower bounds on this utility as a function of a parameter $\delta \in (0,1)$, which quantifies the agent's best-response suboptimality. These bounds highlight the *utility cost of robustness*, since δ can be interpreted as a measure of the robustness level of the principal's contracts. Interestingly, our results show that, unlike in general Stackelberg games (see, *e.g.*, [Gan et al., 2023a]), these bounds do *not* depend on the inducibility gap $\Delta > 0$ characterizing the problem instance.

To derive the bounds, we prove that any non-robust contract can be converted into a robust one by appropriately shifting its payments in the direction of the principal's rewards, and that in an optimal robust contract—provided it yields the principal a strictly positive utility—the agent's utility must be at least δ. We first establish upper and lower bounds that identify the range of possible values of $\mathrm{OPT}(\delta)$. These bounds depend only on the parameter $\delta \in (0,1)$, the value of an optimal non-robust contract OPT, and the social welfare SW achievable with non-robust contracts.

Proposition 8.1 (Upper and lower bounds)**.** *Given an instance of hidden-action principal-agent problem:*

1. *For every $\delta \in (0,1)$, it holds:*

$$\mathrm{OPT}(\delta) \geq \mathrm{OPT}_{\mathrm{LB}}(\delta) \coloneqq \mathrm{OPT} - 2\sqrt{\delta} + \delta.$$

2. *For every $\delta \in (0,1)$, it holds:*

$$\mathrm{OPT}(\delta) \leq \mathrm{OPT}_{\mathrm{UB}}(\delta) \coloneqq \max\left(0,\, \mathrm{SW} - \delta\right).$$

Proof. We prove the two points separately.

1. Let $p' = (1-\sqrt{\delta})p + \sqrt{\delta}r$ for any contract $p \in \mathbb{R}^m_+$. We start by observing that the following inequality holds:

$$\sum_{\omega\in\Omega} F_{a(p),\omega} p_\omega - c_{a(p)} \geq \sum_{\omega\in\Omega} F_{a^\delta(p'),\omega} p_\omega - c_{a^\delta(p')},$$

because $a(p) \in A(p)$. Furthermore, we have:

$$\sum_{\omega\in\Omega} F_{a^\delta(p'),\omega} p'_\omega - c_{a^\delta(p')} > \max_{a'\in\mathcal{A}} \sum_{\omega\in\Omega} F_{a',\omega} p_\omega - c_{a'} - \delta \geq \sum_{\omega\in\Omega} F_{a(p),\omega} p'_\omega - c_{a(p)} - \delta,$$

thanks to the definition of $A^\delta(p')$. Then, by employing the two above inequalities and the definition of p', we have:

$$\begin{aligned}
\delta &\geq \sum_{\omega\in\Omega} F_{a(p),\omega} p'_\omega - c_{a(p)} - \left(\sum_{\omega\in\Omega} F_{a^\delta(p'),\omega} p'_\omega - c_{a^\delta(p')}\right) \\
&= \sum_{\omega\in\Omega} \Big(F_{a(p),\omega} p_\omega - c_{a(p)} - F_{a^\delta(p'),\omega} p_\omega - c_{a^\delta(p')} + \sqrt{\delta}\left(F_{a(p),\omega} - F_{a^\delta(p'),\omega}\right)(r_\omega - p_\omega)\Big) \\
&\geq \sqrt{\delta} \sum_{\omega\in\Omega} \left(F_{a(p),\omega} - F_{a^\delta(p'),\omega}\right)(r_\omega - p_\omega).
\end{aligned}$$

Thus, by rearranging the latter inequality we get:

$$\sqrt{\delta} \geq \left(\sum_{\omega\in\Omega} \left(F_{a(p),\omega} - F_{a^\delta(p'),\omega}\right)(r_\omega - p_\omega)\right). \tag{8.2}$$

Finally, we have

$$\begin{aligned} u^{\mathsf{P}}(p, a^\star(p)) - u^{\mathsf{P}}(p', a^\delta(p')) &= \sum_{\omega\in\Omega} F_{a(p),\omega}(r_\omega - p_\omega) - \left(\sum_{\omega\in\Omega} F_{a^\delta(p'),\omega}(r_\omega - p'_\omega)\right) \\ &= \sum_{\omega\in\Omega} F_{a(p),\omega}(r_\omega - p_\omega) - \left(1-\sqrt{\delta}\right)\left(\sum_{\omega\in\Omega} F_{a^\delta(p'),\omega}(r_\omega - p_\omega)\right) \\ &\leq \left(1-\sqrt{\delta}\right)\left(\sum_{\omega\in\Omega} (F_{a(p),\omega} - F_{a^\delta(p'),\omega})(r_\omega - p_\omega)\right) + \sqrt{\delta} \\ &\leq 2\sqrt{\delta} - \delta, \end{aligned}$$

where the second equality holds thanks to the definition of p', the first inequality because $u^{\mathsf{P}}(p, a(p)) \leq 1$ for each $p \in \mathbb{R}^m_+$ and the second inequality because of Equation (8.2). Finally, let $p \in \mathbb{R}$ be an optimal (non-robust) contract, then we have:

$$\text{OPT} - 2\sqrt{\delta} + \delta \leq u^{\mathsf{P}}(p', a^\delta(p')) \leq \text{OPT}(\delta),$$

concluding the first part of the proof.

2. We split the proof into two parts.

   (a) if $\text{OPT}(\delta) = 0$. Then, we trivially have :

   $$\begin{aligned} 0 = \text{OPT}(\delta) &\leq \max\left(0, \max_{a\in\mathcal{A}} \sum_{\omega\in\Omega} F_{a,\omega} r_\omega - c_a - \delta\right) \\ &= \max\left(0, \text{SW} - \delta\right) \end{aligned}$$

   (b) if $\text{OPT}(\delta) > 0$. We define $p^\star \in \mathbb{R}^m_+$ as an optimal δ-robust contract. Then, we notice that $a_1 \notin \mathcal{A}^\delta(p^\star)$, where a_1 is the opt-out action. Indeed, if $a_1 \in \mathcal{A}^\delta(p^\star)$, then the agent may select the action a_1 as a δ-robust best-response, which provides zero or negative utility to the principal and contradicts the fact that $\text{OPT}(\delta) > 0$. Therefore, in an optimal robust contract $p^\star$, we must have:

   $$\sum_{\omega\in\Omega} F_{a',\omega} p^\star_\omega - c_{a'} \geq \sum_{\omega\in\Omega} F_{a_1,\omega} p^\star_\omega - c_{a_1} + \delta \geq \sum_{\omega\in\Omega} F_{a_1,\omega} p^\star_\omega + \delta \geq \delta,$$

   for some $a' \in \mathcal{A}^\delta(p^\star)$. Thus, we have:

   $$\begin{aligned} \text{OPT}(\delta) &= \sum_{\omega\in\Omega} F_{a^\delta(p),\omega}(r_\omega - p^\star_\omega) \\ &\leq \sum_{\omega\in\Omega} F_{a',\omega}(r_\omega - p^\star_\omega) \end{aligned}$$

$$\leq \sum_{\omega\in\Omega} F_{a',\omega} r_\omega - c_{a'} - \delta$$
$$\leq \max_{a\in\mathcal{A}} \sum_{\omega\in\Omega} F_{a,\omega} r_\omega - c_a - \delta$$
$$\leq \max\left(0,\, \max_{a\in\mathcal{A}} \sum_{\omega\in\Omega} F_{a,\omega} r_\omega - c_a - \delta\right) = \max\left(0,\, \mathrm{SW} - \delta\right).$$

The first inequality above follows from the fact that $a' \in \mathcal{A}^\delta(p^\star)$, while the second inequality holds due to the previous observation.

The two above points conclude the proof. □

In order to prove the first point in Proposition 8.1, we show that it is always possible to convert a non-robust contract into a robust one by suitably moving it towards the direction of the principal's reward vector. In order to prove the second point of Proposition 8.1, we observe that, if an optimal δ-robust contract $p^\star$ provides the principal with an expected utility strictly larger than zero, then the opt-out action does *not* belong to the set of δ-best responses for such a contract. Therefore, in an optimal δ-robust contract that provides the principal with an expected utility strictly larger than zero, the agent's utility is at least $\delta > 0$, and thus the principal's expected utility is at most $\mathrm{SW} - \delta$.

The following proposition shows that the upper and lower bounds in Proposition 8.1 are tight. Formally:

Proposition 8.2. *Given $\delta \in (0,1)$, for every integer $n > n(\delta)$, there is an instance of hidden-action principal-agent problem (parametrized by δ) with $2n+1$ actions, where:*

$$\mathrm{OPT}(\delta) - \mathrm{OPT}_{\mathrm{LB}}(\delta) \leq \mathcal{O}\left(\frac{1}{n}\right).$$

Furthermore, for any $\delta \in (0,1)$, there exists an instance of hidden-action principal-agent problem in which it holds:

$$\mathrm{OPT}(\delta) = \mathrm{OPT}_{\mathrm{UB}}(\delta).$$

Proof. We prove the two points separately.

1. We consider an instance parametrized by $\delta > 0$, with $|\Omega| = 2$ and $|\mathcal{A}| = 2n+1$ for some $n \in \mathbb{N}$, to be defined below. We let:

$$\kappa = \min\left\{ i \in \mathbb{N}_{>0} \,|\, \sqrt{\delta} < \frac{i-1}{i} \right\}.$$

Furthermore, we introduce the following decreasing sequence:

$$\gamma_i = \begin{cases} \frac{i}{i-1} & i = \kappa, \dots, n \\ \frac{2n+1-i}{2n+2-i} & i = n+2, \dots, 2n, \end{cases}$$

with $\gamma_{n+1} = 1$ and $\gamma_{2n+1} = 0$, where $n \in \mathbb{N}$ is such that $n > \kappa$. The distributions over the set of outcomes for the different actions and their corresponding costs are given by:

$$\begin{cases} F_{a_i,\omega_1} = 0 & c_{a_i} = 0,\ i = 1, \dots, \kappa - 1. \\ F_{a_i,\omega_1} = 1 - \gamma_i\sqrt{\delta} & c_{a_i} = 0,\ i = \kappa, \dots, n. \\ F_{a_{2n+1},\omega_1} = 1 & c_{a_{2n+1}} = 0. \end{cases}$$

and the principal's reward is $r = (1, 0)$. Notice that the distribution over outcomes of the different actions are always well defined because of the definition of $\kappa > 0$.

Furthermore, since all the agent's actions a_i with $i < k$ are coincident, we assume for the sake of presentation that the agent always selects a_1. It is easy to verify that the value of an optimal (non-robust) contract is $\text{OPT} = 1$ since the agent breaks ties optimistically and $c_{2n+1} = 0$.

With a similar argument to the one proposed by Dütting et al. [2024] in Proposition 3.9, an optimal δ-robust contract is such that $p^\star = (0, \alpha)$ for some $\alpha \in \mathbb{R}_+$. Consequently, in the rest of the proof, we focus on determining the maximum utility achievable in a linear, δ-robust contract.

For every $a_i \in \mathcal{A}$, with $\kappa < i \leq 2n + 1$, if $\alpha \in \mathbb{R}_+$ is such that $a^\delta(\alpha r) = a_i$, then the two following conditions hold.

(a) All the actions $a_j \notin \mathcal{A}^\delta(\alpha r)$ for each $j < i$. This is because

$$u^{\mathsf{P}}(\alpha r, a_j) \leq u^{\mathsf{P}}(\alpha r, a_i)$$

for each $\alpha \in \mathbb{R}_+$ since $\{\gamma_i\}_{i\in[2n+1]}$ is a decreasing sequence. Thus, the latter observation implies that:

$$\begin{aligned} u^{\mathsf{A}}(\alpha r, a_j) \leq u^{\mathsf{A}}(\alpha r, a_{i-1}) &= \alpha R_{a_{i-1}} - c_{a_{i-1}} \\ &\leq \max_{i\in[2n+1]} u^{\mathsf{A}}(\alpha r, a_i) - \delta \\ &= u^{\mathsf{A}}(\alpha r, a_{2n+1}) - \delta \\ &= \alpha R_{a_{2n+1}} - c_{a_{2n+1}} - \delta \\ &= \alpha - \delta. \end{aligned}$$

Therefore, we have that:

$$\alpha R_{i-1} \leq \alpha - \delta,$$

and, thus, $\alpha \geq \sqrt{\delta}/\gamma_{i-1}$.

(b) The action $a_i \in \mathcal{A}^\delta(\alpha r)$. Thus,

$$\alpha R_{a_i} - c_{a_i} > \max_{i \in [2n+1]} u^{\mathsf{A}}(\alpha r, a_i) - \delta = \alpha - \delta,$$

which implies that $\alpha < \sqrt{\delta}/\gamma_i$.

The two observation above shows that a_i, with $i > \kappa$, is a δ-best response for all the values of α such that:

$$\sqrt{\delta}/\gamma_{i-1} \leq \alpha < \sqrt{\delta}/\gamma_i.$$

Thus, the smallest value of $\alpha \in \mathbb{R}_+$ such that $a^\delta(\alpha r) = a_i$ is $\alpha_i := \sqrt{\delta}/\gamma_{i-1}$, when $i > \kappa$. With the same argument above, it is possible to show that $a_1 = a^\delta(\alpha r)$ for all $\alpha \in [\alpha_1, \alpha_\kappa)$ and $a_\kappa = a^\delta(\alpha r)$ for all $\alpha \in [\alpha_\kappa, \alpha_{\kappa+1})$, with $\alpha_1 = 0$ and $\alpha_\kappa = \delta$.

We observe that the principal's utility in each α_i, with $i \in \{\kappa+1, ..., 2n+1\}$, is such that:

$$\begin{aligned} u^{\mathsf{P}}(\alpha_i r, a_i) = (1-\alpha_i)R_{a_i} &= \left(1 - \frac{\sqrt{\delta}}{\gamma_{i-1}}\right)\left(1 - \gamma_i\sqrt{\delta}\right) \\ &= \left(1 - \frac{\sqrt{\delta}}{\gamma_{i-1}}\right)\left(1 - \gamma_i\sqrt{\delta}\right) \\ &= \left(1 - \left(\gamma_i + \frac{1}{\gamma_{i-1}}\right)\right)\sqrt{\delta} + \frac{\gamma_i}{\gamma_{i-1}}\delta. \end{aligned}$$

Therefore, using the latter formula, we have that the following holds.

(a) If $i = \kappa + 1, \ldots, n$, we have:

$$u^{\mathsf{P}}(\alpha_i r, a_i) \leq 1 - 2\sqrt{\delta} + \frac{\gamma_i}{\gamma_{i-1}}\delta \leq 1 - 2\sqrt{\delta} + \delta,$$

since $\{\gamma_i\}_{i \geq \kappa}$ is a decreasing sequence.

(b) If $i = n+1, n+2$, we have:

$$u^{\mathsf{P}}(\alpha_i r, a_i) \leq 1 - \left(1 + \frac{n-1}{n}\right)\sqrt{\delta} + \frac{n-1}{n}\delta \leq 1 - 2\sqrt{\delta} + \delta + \frac{\sqrt{\delta}}{n}.$$

(c) If $i = n+3, \ldots, 2n+1$, we have:

$$u^{\mathsf{P}}(\alpha_i r, a_i) \leq 1 - 2\sqrt{\delta} + \frac{\gamma_i}{\gamma_{i-1}}\delta \leq 1 - 2\sqrt{\delta} + \delta,$$

since $\{\gamma_i\}_{i \geq \kappa}$ is a decreasing sequence.

We also observe that the principal's utility in α_κ is such that:

$$u^{\mathsf{P}}(\alpha_\kappa r, a_k) = (1-\delta)(1-\gamma_\kappa\sqrt{\delta}).$$

since $\gamma_k > 1$ and $\delta > 0$. Thanks to the definition of κ, with a simple calculation, it possible to show that:

$$\kappa = \begin{cases} \left\lceil \frac{1}{1-\sqrt{\delta}} \right\rceil & \text{if } \frac{1}{1-\sqrt{\delta}} \notin \mathbb{N} \\ \frac{1}{1-\sqrt{\delta}} + 1 & \text{if } \frac{1}{1-\sqrt{\delta}} \in \mathbb{N}. \end{cases}$$

Thus, when $1/(1-\sqrt{\delta}) \notin \mathbb{N}$, we have:

$$\begin{aligned} u^{\mathsf{P}}(\alpha_\kappa r, a_k) &= (1-\delta)(1-\gamma_\kappa\sqrt{\delta}) \\ &= (1-\delta)\left(1 - \frac{\left\lceil \frac{1}{1-\sqrt{\delta}} \right\rceil}{\left\lceil \frac{1}{1-\sqrt{\delta}} \right\rceil - 1}\sqrt{\delta}\right) \\ &\leq (1-\delta)\left(1 - 2\sqrt{\delta} + \delta\right). \end{aligned}$$

Similarly, it is possible to show that the same relation holds even when $1/(1-\sqrt{\delta}) \in \mathbb{N}$. Finally, by putting all together, we have that:

$$\begin{aligned} \mathrm{OPT}(\delta) - \mathrm{OPT}_{\mathrm{LB}}(\delta) &\leq \max_{i \in [2n+1]} u^{\mathsf{P}}(\alpha_i r, a_i) - (1 - 2\sqrt{\delta} + \delta) \\ &= u^{\mathsf{P}}(\alpha_{n+1} r, a_{n+1}) - (1 - 2\sqrt{\delta} + \delta) \leq \frac{\sqrt{\delta}}{n}, \end{aligned}$$

concluding the first part of the proof.

2. We consider an instance with $|\Omega| = |\mathcal{A}| = 2$ and $r = (0, 1)$. The distributions over the set of outcomes for the different actions and their corresponding costs are given by:

$$\begin{cases} F_{a_1} = (1, 0) & c_{a_1} = 0 \\ F_{a_2} = (0, 1) & c_{a_2} = 0. \end{cases}$$

It is easy to verify that $\mathrm{SW} = R_{a_2} - c_{a_2} = 1$. Furthermore, with a similar argument to the one proposed by Dütting et al. [2024] in Proposition 3.9, an optimal δ-robust contract is such that $p^\star = (0, \alpha)$ for some $\alpha \in \mathbb{R}_+$.

We show that $\mathrm{OPT}(\delta) = 1 - \delta$. Let $a_1 \in \mathcal{A}$ be the opt-out action. We observe that $a_1 \notin A^\delta(\alpha r)$ for all the $\alpha \in \mathbb{R}_+$ that satisfy:

$$\alpha R_{a_2} - c_{a_2} = \alpha \geq \alpha R_{a_1} - c_{a_1} + \delta = \delta.$$

Thus, $a_1 \notin A^\delta(\alpha r)$ for all $\alpha \geq \delta$. Therefore, the smallest $\alpha \in \mathbb{R}_+$ such that the agent selects the action a_2 coincides with $\delta > 0$. Thus, the largest utility the principal can achieve satisfies $\mathrm{OPT}(\delta) = 1 - \delta = \mathrm{SW} - \delta$.

The two points above conclude the proof. □

Proposition 8.2 shows that our upper bound $\mathrm{OPT}_{\mathrm{UB}}(\delta)$ is (strictly) tight, while our lower bound $\mathrm{OPT}_{\mathrm{LB}}(\delta)$ is tight up to an additive term that linearly goes to zero

as the number of agent's actions n increases.

Finally, thanks to the previous propositions, we can show the following property about the value of an optimal δ-robust contracts as a function of $\delta \in (0,1)$.

Proposition 8.3. *The function $(0,1) \ni \delta \mapsto \mathrm{OPT}(\delta)$ is continuous and non-increasing in δ. Moreover, $\lim_{\delta\to 0^+} \mathrm{OPT}(\delta) = \mathrm{OPT}$ and $\lim_{\delta\to 1^-} \mathrm{OPT}(\delta) = 0$.*

Proof. We prove the claims separately.

1. We first show that $\mathrm{OPT}(\delta)$ is a continuous function. Let $p^\star \in \mathbb{R}^m_+$ be a δ-robust and $p' = (1-\sqrt{\epsilon})p^\star + \sqrt{\epsilon} r$. Then, by Lemma 8.1, we have:

$$\begin{aligned}\mathrm{OPT}(\delta+\epsilon) &\geq u^{\mathrm{P}}(p', a^{\delta+\epsilon}(p')) \\ &\geq u^{\mathrm{P}}(p^\star, a^{\delta}(p^\star)) - 2\sqrt{\epsilon} + \epsilon \\ &= \mathrm{OPT}(\delta) - 2\sqrt{\epsilon} + \epsilon.\end{aligned}$$

   Thus, rewriting the latter inequality and taking the limit, we have that:

$$0 = \lim_{\epsilon\to 0}(2\sqrt{\epsilon} - \epsilon) \geq \lim_{\epsilon\to 0} \mathrm{OPT}(\delta) - \mathrm{OPT}(\delta+\epsilon).$$

   Let $p^\star \in \mathbb{R}^m_+$ be a δ'-robust and $p' = (1-\sqrt{\epsilon})p^\star + \sqrt{\epsilon} r$ with $\delta' = \delta - \epsilon$. Then, by Lemma 8.1, we have:

$$\begin{aligned}\mathrm{OPT}(\delta) &= \mathrm{OPT}(\delta'+\epsilon) \\ &\geq u^{\mathrm{P}}(p', a^{\delta'+\epsilon}(p')) \\ &\geq u^{\mathrm{P}}(p^\star, a^{\delta'}(p^\star)) - 2\sqrt{\epsilon} + \epsilon \\ &= \mathrm{OPT}(\delta-\epsilon) - 2\sqrt{\epsilon} + \epsilon.\end{aligned}$$

   Thus, by taking the limit, we have that:

$$0 = \lim_{\epsilon\to 0}(2\sqrt{\epsilon} - \epsilon) \geq \lim_{\epsilon\to 0} \mathrm{OPT}(\delta-\epsilon) - \mathrm{OPT}(\delta).$$

   Since the latter considerations hold for every $\delta \in (0,1)$, we can easily show that $\mathrm{OPT}(\delta)$ is continuous in $\delta \in (0,1)$.

2. We now prove that $\mathrm{OPT}(\delta)$ is non-increasing. Let $\delta, \delta' \in (0,1)$ such that $\delta' < \delta$. Furthermore, let $p^\star \in \mathbb{R}^m_+$ be an optimal δ-robust contract. Therefore, we have:

$$\mathrm{OPT}(\delta) = u(p^\star, a^{\delta}(p^\star)) \leq u(p^\star, a^{\delta'}(p^\star)) \leq \mathrm{OPT}(\delta'),$$

   since, for each $p \in \mathbb{R}^m_+$, we have $A^{\delta'}(p) \subseteq A^{\delta}(p)$.

3. By Proposition 8.1, we have:

$$\lim_{\delta \to 0^+} \mathrm{OPT}(\delta) \geq \lim_{\delta \to 0^+} \mathrm{OPT} - 2\sqrt{\delta} + \delta = \mathrm{OPT},$$

and,

$$\lim_{\delta \to 1^-} \mathrm{OPT}(\delta) \leq \lim_{\delta \to 1^-} 1 - \delta = 0.$$

The two points above conclude the proof. □

Figure 7.1 shows an example of the region in which the value of $\mathrm{OPT}(\delta)$ is bounded, as defined by the lower and upper bounds in Propositions 8.1 and 8.3

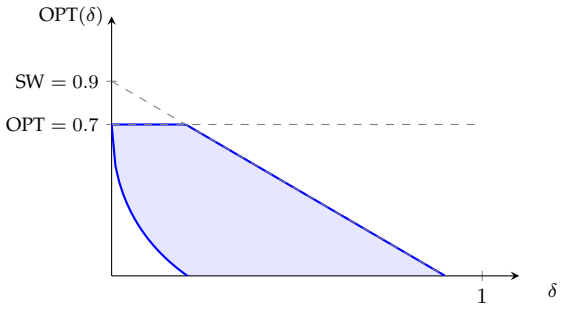

Figure 8.1: *The blue area corresponds to the region in which* $\mathrm{OPT}(\delta)$ *is bounded as a function of* $\delta \in (0, 1)$ *in an instance of hidden-action principal-agent problem with* $\mathrm{SW} = 0.9$ *and* $\mathrm{OPT} = 0.7$.

Comparison with Stackelberg games. The characterization results derived in this section exhibit some crucial differences compared to analogous results derived for Stackelberg games by Gan et al. [2023a]. Indeed, in such settings, Gan et al. [2023a] show that the value of an optimal δ-robust commitment crucially depends on a parameter $\Delta > 0$ that represents the *inducibility gap* of the problem instance. Intuitively, the inducibility gap encodes how easy it is for the leader to induce the follower to play *any* action; see [Gan et al., 2023a] for a formal definition. Specifically, in Stackelberg games, the leader's expected utility $\mathrm{OPT}(\delta)$ in an optimal δ-robust commitment is a Lipschitz function lower bounded by $\mathrm{OPT} - \delta/\Delta$ if $\delta < \Delta$, whereas $\mathrm{OPT}(\delta)$ may *not* be even a continuous function if $\delta > \Delta$. In contrast, in hidden-action principal-agent problems, the value of an optimal δ-robust contract is a continuous function with respect to $\delta \in (0, 1)$, regardless of the inducibility gap of the instance. Furthermore, the value of $\mathrm{OPT}(\delta)$ is either zero or it is upper bounded by $\mathrm{SW} - \delta$, showing that for large values of δ, the maximum utility that the principal can achieve may be particularly small. Intuitively, this is because the principal must provide the agent with a large expected payment to induce them to take desirable actions rather than the opt-out one. This upper bound does *not* generally hold in Stackelberg games in which the principal may achieve a large utility even for large values of δ.

8.3 Online learning of robust contracts

In this section, we study the online learnability of robust contracts and propose an algorithm that achieves sublinear regret. To this end, we show that it is possible to employ a simpler discretization of the contract space that does not require *a priori* knowledge of the principal's reward function.

8.3.1 Learning interaction

We consider an online learning framework that generalizes the one presented in Chapter 6, in which the features of agent's actions, *i.e.*, costs and probabilities over outcomes, depend on an agent's *type* that is sampled at each round from some (fixed) unknown probability distribution. Before formally defining the online learning framework, we introduce some notation that is needed to deal with hidden-action principal-agent problems in which the agent can be of different types.

Hidden-action principal-agent problems with types. We let Θ be the finite set of possible agent's types, while $F_{\theta,a}$ and $c_{\theta,a}$ denote the probability distribution over outcomes and the cost, respectively, of action $a \in A$, when the agent has type $\theta \in \Theta$. The agent's type is drawn from an unknown probability distribution $\lambda \in \Delta_\Theta$, with $\lambda_\theta \in [0,1]$ being the probability of type $\theta \in \Theta$. We extend all the notation introduced in Chapter 2 to account for agent's types. Specifically, given $\delta \in (0,1)$, we let $A^{\theta,\delta}(p) \subseteq \mathcal{A}$ be the set of δ-best responses to a contract $p \in \mathbb{R}^m_+$ for type $\theta \in \Theta$. Moreover, we denote by $a^{\theta,\delta}(p)$ the δ-best response that is played by the agent under a δ-robust contract, namely the worst one for the principal. Formally,

$$a^{\theta,\delta}(p) \in \arg\min_{a \in A^\delta(p)} \sum_{\omega \in \Omega} F_{\omega,\theta,a} \cdot (r_\omega - p_\omega)$$

Similarly, we use $A^\theta(p)$ and $a^\theta(p)$ for (exact) best responses. Finally, the principal's expected utility when committing to a contract $p \in \mathbb{R}^m_+$ against an agent of type $\theta \in \Theta$ playing an action $a \in \mathcal{A}$ is

$$u^{\mathsf{P}}(p,a,\theta) \coloneqq \sum_{\omega \in \Omega} F_{\omega,\theta,a} \cdot (r_\omega - p_\omega).$$

We study an online learning framework in which the principal and the agent repeatedly interact over $T > 0$ rounds. Each round involves a repetition of the same instance of hidden-action principal-agent problem, with only the agent's type changing from round to round. Specifically, at each round $t \in [T]$, the principal-agent interaction is as follows:

1. The agent's type $\theta^t \in \Theta$ is sampled from the distribution λ. Notice that θ^t is *not* observed by the principal.

2. The principal commits to a contract $p^t \in \mathcal{C} \coloneqq [0,1]^m$.

3. After observing the contract p^t, the agent plays a δ-best response $a^t \coloneqq a^{\theta^t,\delta}(p^t)$. Notice that the action a^t is *not* observed by the principal.

4. The principal observes the outcome $\omega^t \sim F_{a^t}$ that is realized as an effect of the agent's action a^t.

We remark that, in the interaction described above, the principal's contract design space is assumed to be limited to the hypercube $\mathcal{C} := [0,1]^m$. Restricting the contract space to a bounded set is standard in the literature on online learning in contract design (see, *e.g.*, Ho et al. [2014], Zhu et al. [2023], Chen et al. [2024]) and it is motivated by the negative result proved in Theorem 1 in Chapter 6. Therefore, we need to introduce a formal definition of the principal's expected utility when committing to an optimal δ-robust contract restricted to such a space. Formally:

$$\mathrm{OPT}(\mathcal{C}, \delta) := \max_{p \in \mathcal{C}} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p, a^{\theta,\delta}(p), \theta).$$

The goal of the principal is to minimize the (*cumulative*) *pseudo-regret*, or simply *regret*, which can be defined as:

$$R_T(\mathcal{C}, \delta) := T \cdot \mathrm{OPT}(\mathcal{C}, \delta) - \mathbb{E}\left[\sum_{t \in [T], \theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\right],$$

where the expectation is over the randomness of the learning algorithm. Our goal is to design a *no-regret* algorithm for the principal, namely one achieving $R_T(\mathcal{C}, \delta) = o(T)$.

8.3.2 A no-regret algorithm

Before introducing our no-regret learning algorithm (Algorithm 8.1), we prove a key lemma about the "continuity" of the principal's expected utility over the hypercube. In particular, given a δ-robust contract $p \in \mathbb{R}^m_+$, we show that it is possible to build another contract $p' \in \mathbb{R}^m_+$ that is $(\delta + \epsilon)$-robust and provides the principal with utility at most $2\sqrt{\epsilon}$ worse than the one of contract p. Formally:

Lemma 8.1. *Given any $\delta, \epsilon \in (0,1)$ and a contract $p \in \mathbb{R}^m_+$, the following holds for every $\theta \in \Theta$:*

$$u^{\mathsf{P}}(p', a^{\theta,\delta+\epsilon}(p'), \theta) \geq u^{\mathsf{P}}(p, a^{\theta,\delta}(p), \theta) - 2\sqrt{\epsilon},$$

where $p' := (1 - \sqrt{\epsilon})p + \sqrt{\epsilon} r$.

Proof. We prove the lemma for a given type $\theta \in \Theta$, omitting the explicit dependence on θ to simplify the notation. Let $\eta = \sqrt{\epsilon}$ and let $p' = (1-\eta)p + \eta r$. For ease of notation, let

$$a = a^{\delta}(p) \qquad \text{and} \qquad b = a^{\delta+\epsilon}(p').$$

We prove that

$$u^{\mathsf{P}}(p', b) \geq u^{\mathsf{P}}(p, a) - 2\sqrt{\epsilon}.$$

We first prove two identities that will be used throughout the proof. For every action $a \in A$,

$$u^{\mathsf{A}}(p', a) = F_a \cdot p' - c_a$$

$$\begin{aligned} &= F_a \cdot \big((1-\eta)p + \eta r\big) - c_a \\ &= F_a \cdot p - c_a + \eta F_a \cdot (r-p) \\ &= u^{\mathsf{A}}(p,a) + \eta u^{\mathsf{P}}(p,a), \end{aligned} \tag{8.3}$$

where the last equality follows from the definitions of u^{A} and u^{P}. Similarly,

$$\begin{aligned} u^{\mathsf{P}}(p',a) &= F_a \cdot (r-p') \\ &= F_a \cdot \big(r - (1-\eta)p - \eta r\big) \\ &= (1-\eta) F_a \cdot (r-p) \\ &= (1-\eta) u^{\mathsf{P}}(p,a). \end{aligned} \tag{8.4}$$

Moreover, since $r \in [0,1]^m$, $p \geq 0$, and $F_a \in \Delta_\Omega$, we have

$$u^{\mathsf{P}}(p,a) = F_a \cdot (r-p) \leq 1.$$

Therefore, by Equation (8.4),

$$\begin{aligned} u^{\mathsf{P}}(p',a) &= (1-\eta) u^{\mathsf{P}}(p,a) \\ &= u^{\mathsf{P}}(p,a) - \eta u^{\mathsf{P}}(p,a) \\ &\geq u^{\mathsf{P}}(p,a) - \eta. \end{aligned} \tag{8.5}$$

We now distinguish two cases.

Case 1: $b \in A^\delta(p)$**.** Since $a = a^\delta(p)$ is chosen as the action in $A^\delta(p)$ minimizing the principal's utility, and since $b \in A^\delta(p)$, we have

$$u^{\mathsf{P}}(p,a) \leq u^{\mathsf{P}}(p,b). \tag{8.6}$$

Using Equation (8.5) and then using Equation (8.6), we obtain

$$\begin{aligned} u^{\mathsf{P}}(p',b) &\geq u^{\mathsf{P}}(p,b) - \eta \\ &\geq u^{\mathsf{P}}(p,a) - \eta \\ &\geq u^{\mathsf{P}}(p,a) - 2\eta. \end{aligned} \tag{8.7}$$

Case 2: $b \notin A^\delta(p)$**.** Let $\bar{a} \in \arg\max_{a \in A} u^{\mathsf{A}}(p,a)$, and define

$$M = \max_{a \in A} u^{\mathsf{A}}(p,a) = u^{\mathsf{A}}(p,\bar{a}).$$

Since $\bar{a}$ is an exact best response under p, we have $\bar{a} \in A^\delta(p)$. Hence, by the definition of $a = a^\delta(p)$,

$$u^{\mathsf{P}}(p,a) \leq u^{\mathsf{P}}(p,\bar{a}). \tag{8.8}$$

Since $b \notin A^\delta(p)$, by the definition of $A^\delta(p)$ we have

$$u^{\mathsf{A}}(p,b) \leq M - \delta. \tag{8.9}$$

On the other hand, since $b = a^{\delta+\epsilon}(p')$, we have $b \in A^{\delta+\epsilon}(p')$. Therefore,

$$u^{\mathsf{A}}(p', b) > \max_{a \in A} u^{\mathsf{A}}(p', a) - \delta - \epsilon.$$

In particular, since

$$\max_{a \in A} u^{\mathsf{A}}(p', a) \geq u^{\mathsf{A}}(p', \bar{a}),$$

we get

$$u^{\mathsf{A}}(p', b) > u^{\mathsf{A}}(p', \bar{a}) - \delta - \epsilon. \tag{8.10}$$

Using Equation (8.3) in Equation (8.10), we obtain

$$u^{\mathsf{A}}(p, b) + \eta u^{\mathsf{P}}(p, b) > u^{\mathsf{A}}(p, \bar{a}) + \eta u^{\mathsf{P}}(p, \bar{a}) - \delta - \epsilon. \tag{8.11}$$

Since $u^{\mathsf{A}}(p, \bar{a}) = M$, Equation (8.11) becomes

$$u^{\mathsf{A}}(p, b) + \eta u^{\mathsf{P}}(p, b) > M + \eta u^{\mathsf{P}}(p, \bar{a}) - \delta - \epsilon.$$

Combining this with Equation (8.9), we get

$$M - \delta + \eta u^{\mathsf{P}}(p, b) > M + \eta u^{\mathsf{P}}(p, \bar{a}) - \delta - \epsilon.$$

Canceling $M - \delta$ from both sides gives

$$\eta\big(u^{\mathsf{P}}(p, \bar{a}) - u^{\mathsf{P}}(p, b)\big) < \epsilon. \tag{8.12}$$

Since $\eta = \sqrt{\epsilon}$, Equation (8.12) implies

$$u^{\mathsf{P}}(p, \bar{a}) - u^{\mathsf{P}}(p, b) < \eta. \tag{8.13}$$

Combining Equation (8.8) and Equation (8.13), we obtain

$$u^{\mathsf{P}}(p, a) - u^{\mathsf{P}}(p, b) < \eta. \tag{8.14}$$

Finally, applying Equation (8.5) and using Equation (8.14), we get

$$\begin{aligned} u^{\mathsf{P}}(p', b) &\geq u^{\mathsf{P}}(p, b) - \eta \\ &> u^{\mathsf{P}}(p, a) - 2\eta. \end{aligned} \tag{8.15}$$

By Equation (8.7) and Equation (8.15), in both cases we have

$$u^{\mathsf{P}}(p', b) \geq u^{\mathsf{P}}(p, a) - 2\eta.$$

Substituting back $a = a^{\delta}(p)$, $b = a^{\delta+\epsilon}(p')$, and $\eta = \sqrt{\epsilon}$, we obtain

$$u^{\mathsf{P}}(p', a^{\delta+\epsilon}(p')) \geq u^{\mathsf{P}}(p, a^{\delta}(p)) - 2\sqrt{\epsilon}.$$

This proves the claim. □

We observe that the idea of shifting payments towards the principal's reward vector, as in Lemma 8.1, was first adopted by Dutting et al. [2021] in non-robust settings, in order to deal with approximately incentive-compatible contracts.

Thanks to Lemma 8.1, it is possible to show that there always exists a contract $p \in \mathcal{B}_\epsilon$ that satisfies the following condition, where $\mathcal{B}_\epsilon \subseteq \mathcal{C}$ is a uniform grid of the

hypercube $\mathcal{C}$, built with step size $\epsilon > 0$ (see also Algorithm 8.1).

$$\sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(p, a^{\theta,\delta}(p), \theta) \geq \mathrm{OPT}(\mathcal{C}, \delta) - \mathcal{O}(\sqrt{\epsilon}).$$

Consequently, by suitably choosing $\epsilon > 0$ as a function of the time horizon T, and by instantiating a no-regret algorithm (such as `UCB1` [Auer et al., 2002]) with the set of arms $\mathcal{B}_\epsilon$, we can upper bound the regret suffered by Algorithm 8.1 as follows.

Theorem 8.1. *The regret suffered by Algorithm 8.1 can be upper bounded as follows:*

$$R_T(\mathcal{C}, \delta) \leq \widetilde{\mathcal{O}}\left(T^{1 - \frac{1}{2(m+1)}}\right).$$

Proof. In the following, we let:

1. $p^\star$ be such that $\sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(p^\star, a^{\theta,\delta}(p^\star), \theta) = \mathrm{OPT}(\mathcal{C}, \delta)$.
2. $p' := (1 - \sqrt{2\epsilon})p^\star + \sqrt{2\epsilon} r$. Notice that $p' \in [0,1]^m$ since both $p, r \in \mathcal{C}$ and $\mathcal{C}$ is convex.
3. $\overline{p} \in \mathcal{B}_\epsilon$ be such that $\|\overline{p} - p'\|_\infty \leq \epsilon$. Notice that there always exists at least once contract $\bar{p}$ satisfying the latter condition because of the definition of $\mathcal{B}_\epsilon$ and the fact that $p' \in [0,1]^m$.

We start by observing that, for each $\theta \in \Theta$, it holds $a^{\theta,\delta}(\overline{p}) \in A^{\theta,\delta+2\epsilon}(p')$. Indeed, we have:

$$\begin{aligned}
\sum_{\omega \in \Omega} F_{\theta, a^{\theta,\delta}(\overline{p}), \omega} p'_\omega - c_{\theta, a^{\theta,\delta}(\overline{p})} &\geq \sum_{\omega \in \Omega} F_{\theta, a^{\theta,\delta}(\overline{p}), \omega} \overline{p}_\omega - c_{\theta, a^{\theta,\delta}(\overline{p})} - \epsilon \\
&> \sum_{\omega \in \Omega} F_{\theta, a, \omega} \overline{p}_\omega - c_{\theta, a} - \epsilon - \delta \\
&\geq \sum_{\omega \in \Omega} F_{\theta, a, \omega} p'_\omega - c_{\theta, a} - 2\epsilon - \delta,
\end{aligned}$$

for every actions $a \in \mathcal{A}$. The above inequalities follows since:

$$\sum_{\omega \in \Omega} F_{\theta, a, \omega}(\overline{p}_\omega - p'_\omega) \leq \|F_{\theta, a}\|_1 \|\overline{p} - p'\|_\infty \leq \epsilon,$$

for every actions $a \in \mathcal{A}$ and $\theta \in \Theta$.

Therefore, we can prove that the following holds:

$$\begin{aligned}
\sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(\overline{p}, a^{\theta,\delta}(\overline{p}), \theta) &\geq \sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(p', a^{\theta,\delta}(\overline{p}), \theta) - \epsilon \\
&\geq \sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(p', a^{\theta,\delta+2\epsilon}(p'), \theta) - \epsilon \\
&\geq \sum_{\theta \in \Theta} \lambda_\theta u^{\mathrm{P}}(p^\star, a^{\theta,\delta}(p^\star), \theta) - 2\sqrt{2\epsilon}
\end{aligned}$$

$$= \mathrm{OPT}(\mathcal{C}, \delta) - 2\sqrt{2\epsilon}, \tag{8.16}$$

where the first inequality above holds because

$$\sum_{\omega \in \Omega} F_{\theta,a,\omega}(\overline{p}_\omega - p'_\omega) \leq \|F_{\theta,a}\|_1 \|\overline{p} - p'\|_\infty \leq \epsilon,$$

for every actions $a \in \mathcal{A}$ and type $\theta \in \Theta$. The second inequality holds because of the definition of δ-best response, while the third inequality follows from Lemma 8.1.

At this point, we can decompose the cumulative regret as follows:

$$\begin{aligned} R_T(\mathcal{C}, \delta) &= T \cdot \mathrm{OPT}(\mathcal{C}, \delta) - \mathbb{E}\Big[\sum_{t=1}^{T} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big] \\ &\leq \underbrace{T \cdot \mathrm{OPT}(\mathcal{C}, \delta) - T \cdot \max_{p \in \mathcal{B}_\epsilon} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p, a^{\theta,\delta}(p), \theta)}_{(a)} \\ &\quad + \underbrace{T \cdot \max_{p \in \mathcal{B}_\epsilon} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p, a^{\theta,\delta}(p), \theta) - \mathbb{E}\Big[\sum_{t \in [T], \theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big]}_{(b)}. \end{aligned} \tag{8.17}$$

We focus on bounding term (a) in Equation (8.17). Let $\bar{p} \in [0,1]^m$ be the contract defined at point 3. Then, we have:

$$\begin{aligned} &T \cdot \mathrm{OPT}(\mathcal{C}, \delta) - T \max_{p \in \mathcal{B}_\epsilon} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta) \\ &\leq T \cdot \mathrm{OPT}(\mathcal{C}, \delta) - T \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(\bar{p}, a^{\theta,\delta}(\bar{p}), \theta) \\ &\leq 2\sqrt{2\epsilon}\, T, \end{aligned}$$

where the last inequality holds because of Equation (8.16). We focus on bounding term (b) in Equation (8.17). By employing the same analysis to prove the regret bound in `UCB1` (see, *e.g.*, Auer et al. [2002]), we have:

$$\begin{aligned} &T \cdot \max_{p \in \mathcal{B}_\epsilon} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p, a^{\theta,\delta}(p), \theta) - \mathbb{E}\Big[\sum_{t \in [T]} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big] \\ &\leq \mathcal{O}\left(\sqrt{T |\mathcal{B}_\epsilon| \log(T)}\right). \end{aligned}$$

Finally, observing that $|\mathcal{B}_\epsilon| \leq \mathcal{O}((1/\epsilon)^m)$, by putting (a) and (b) in Equation (8.17) together and setting $\epsilon = T^{-\frac{1}{m+1}}$, we have:

$$R_T(\mathcal{C}, \delta) \leq \widetilde{\mathcal{O}}\left(T^{1 - \frac{1}{2(m+1)}}\right),$$

concluding the proof. □

We remark that the regret lower bound introduced by Zhu et al. [2023] also holds in our setting. This is because, when the parameter δ is arbitrarily small, if we consider the same instances used by Zhu et al. [2023] in their lower bound, the principal is still required to enumerate an exponential number of regions, thus suffering $\Omega(T^{1-1/(m+2)})$ regret. This confirms that an exponential dependence on the number of outcomes is unavoidable in the regret suffered by any algorithm. Furthermore, Theorem 8.1 shows regret guarantees similar to those obtained by Zhu et al. [2023] in the non-robust version of the problem. Indeed, they achieve an upper bound of the order of $\widetilde{\mathcal{O}}(\sqrt{m} \cdot T^{1-1/(2m+1)})$ on the regret.

Algorithm 8.1 Regret minimizer for δ-robust contracts

Require: $T > 0$

1: Set $\epsilon \leftarrow T^{-\frac{1}{m+1}}$
2: $\mathcal{B}_\epsilon \leftarrow \{p \in [0,1]^m \mid p_\omega \in \{0, \epsilon, 2\epsilon, ..., 1\} \, \forall \omega \in \Omega\}$
3: Run `UCB1` with $\mathcal{B}_\epsilon$ as set of arms

Finally, we also show that, when the parameter $\delta \in (0,1)$ is sufficiently small (with respect to the time horizon $T > 0$), Algorithm 8.1 achieves sublinear regret with respect to an optimal non-robust contract within $\mathcal{C}$. Formally, we define the value of such a contract as follows:

$$\text{OPT}(\mathcal{C}) := \max_{p \in \mathcal{C}} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p, a^\theta(p), \theta).$$

Furthermore, when $\text{OPT}(\mathcal{C})$ is chosen as baseline, the regret definition becomes the following:

$$R_T(\mathcal{C}) := T \, \text{OPT}(\mathcal{C}) - \mathbb{E}\left[\sum_{t \in [T], \theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\right],$$

where the expectation is over the randomness of the learning algorithm. We remark that the regret definition above coincides with the one introduced by Zhu et al. [2023]. Then, we can prove that the following corollary of Theorem 8.1 holds.

Corollay 8.1. *The regret suffered by Algorithm 8.1 can be upper bounded as follows:*

$$R_T(\mathcal{C}) \leq \widetilde{\mathcal{O}}\left(T^{1-\frac{1}{2(m+1)}}\right) + 2\sqrt{\delta} T.$$

Proof. Let $p^\star$ be an optimal non-robust contract inside $\mathcal{C}$. Then we have:

$$\begin{aligned} R_T(\mathcal{C}) &= T \cdot \text{OPT}(\mathcal{C}) - \mathbb{E}\Big[\sum_{t=1}^{T} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big] \\ &= T \cdot (\text{OPT}(\mathcal{C}) - \text{OPT}(\mathcal{C}, \delta)) + T \cdot \text{OPT}(\mathcal{C}, \delta) - \mathbb{E}\Big[\sum_{t=1}^{T} \sum_{\theta \in \Theta} \lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big] \end{aligned}$$

$$\leq 2\sqrt{\delta}T + T \cdot \text{OPT}(\mathcal{C}, \delta) - \mathbb{E}\Big[\sum_{t=1}^{T}\sum_{\theta\in\Theta}\lambda_\theta u^{\mathsf{P}}(p^t, a^{\theta,\delta}(p^t), \theta)\Big]$$
$$\leq 2\sqrt{\delta}T + \widetilde{\mathcal{O}}\left(T^{1-\frac{1}{2(m+1)}}\right),$$

where the first inequality holds by employing the same argument needed to prove Proposition 8.1, and the second inequality holds thanks to Theorem 8.1, concluding the proof. □

We remark that, if we set $\delta = 1/T^\alpha$ with $\alpha > 0$ in Corollary 8.1, then the regret with respect to an optimal non-robust contract within the hypercube $\mathcal{C}$ is sublinear.

Remark 8.1. *Our algorithm can be extended to deal with settings in which the agent can play* any *δ-best response within the set $A^{\theta^t,\delta}(p^t)$. In such a setting, the principal's utility is* not *fully stochastic. However, our Algorithm 8.1 can be easily extended by instantiating an adversarial no-regret algorithm instead of* `UCB1`.

Comparison with [Zhu et al., 2023]. We observe that our algorithm provides some advantages compared to the state-of-the-art algorithm proposed by Zhu et al. [2023]. First, our approach employs as set of contracts a simple discretization of the hypercube, while the approach by Zhu et al. [2023] requires determining the minimum set of contracts $\mathcal{V}_\epsilon(\mathcal{C})$—for some suitable $\epsilon > 0$—such that, for every $p \in \mathcal{C}$, there exists a $p' \in \mathcal{V}_\epsilon(\mathcal{C})$ that satisfies $\sum_{\omega\in\Omega}(r_\omega - p_\omega)p'_\omega \geq \cos(\epsilon)$. However, the need of designing such a set of contracts makes the approach by Zhu et al. [2023] challenging to be employed in practice compared to ours. Second, our algorithm does *not* require *apriori* knowledge of the principal's reward, which is instead required by the algorithm of Zhu et al. [2023].

8.4 Computational aspects of robust contracts

In this section, we focus on computing robust contracts that maximize the principal's utility. Our main result is a polynomial-time algorithm for this task, in contrast with the hardness results known in Stackelberg games and Bayesian persuasion. The key idea is to exploit the structure of the robustness constraints. Indeed, once the agent's best response and worst approximate best response are fixed, the problem reduces to solving a collection of $n+1$ linear programs. Thus, by iterating over all possible action pairs, we obtain an optimal robust contract.

Characterizing an optimal contract. We begin by presenting an optimization problem that characterizes an optimal δ-robust contract. Consider an arbitrary optimal δ-robust contract $p^\star$, as well as two arbitrary actions $a^\star \in A(p^\star)$ and $a^\delta \in \arg\min_{a\in A^\delta(p^\star)} u^{\mathsf{P}}(p^\star, a)$. By fixing $a^\star$ and a^δ, we show that the following optimization problem, over the variable $p \in \mathbb{R}^m_+$, characterizes an optimal δ-robust contract $p^\star$.

$$\max_{p\in\mathbb{R}^m_+} \quad u^{\mathsf{P}}(p, a^\delta) \tag{8.18}$$

subject to the following *disjunctive* constraints, which must hold for every agent's action $a \in \mathcal{A}$:

$$\Big(u^{\mathsf{A}}(p,a) \leq u^{\mathsf{A}}(p,a^\star) - \delta\Big) \vee \Big(u^{\mathsf{P}}(p,a) \geq u^{\mathsf{P}}(p,a^\delta)\Big). \tag{8.18a}$$

Intuitively, Equation (8.18a) requires that each action a is either *not* a δ-best response (first inequality) or no worse than a^δ for the principal (second inequality). This ensures that the objective function $u^{\mathsf{P}}(p,a^\delta)$ captures the principal's utility in a δ-robust contract p. More formally, we establish the following lemma (recall that $\Psi(p) = \min_{a\in A^\delta(p)} u^{\mathsf{P}}(p,a)$ denotes the principal's utility in a δ-robust contract p).

Lemma 8.2. *Every optimal solution $p \in \mathbb{R}^m_+$ to Problem (8.18) is an optimal δ-robust contract, i.e., $\Psi(p) = \Psi(p^\star)$.*

Proof. First, observe that $p^\star$ is a feasible solution to Problem (8.18). Indeed, if an action a is *not* a δ-best response to $p^\star$, then by definition this means $u^{\mathsf{A}}(p^\star,a) \leq u^{\mathsf{A}}(p^\star,a^\star) - \delta$; otherwise, it must be that $u^{\mathsf{P}}(p^\star,a) \geq u^{\mathsf{P}}(p^\star,a^\delta)$ as a^δ is by definition the worst δ-best response in $p^\star$. As a result, Equation (8.18a) holds for $p^\star$ for every $a \in \mathcal{A}$.

Furthermore, notice that, according to the definition of a^δ:

$$\Psi(p^\star) = \min_{a\in A_\delta(p^\star)} u^{\mathsf{P}}(p^\star,a) = u^{\mathsf{P}}(p^\star,a^\delta).$$

In order to complete the proof, it is sufficient to show that $u^{\mathsf{P}}(p,a^\delta) \leq \Psi(p)$ for every feasible solution $p \in \mathbb{R}^m_+$ to Problem (8.18). Indeed, if the condition above holds, for any arbitrary optimal solution $p' \in \mathbb{R}^m_+$ to Problem (8.18), we have:

$$\Psi(p^\star) = u^{\mathsf{P}}(p^\star,a^\delta) \leq u^{\mathsf{P}}(p',a^\delta) \leq \Psi(p') \leq \Psi(p^\star),$$

where $u^{\mathsf{P}}(p^\star,a^\delta) \leq u^{\mathsf{P}}(p',a^\delta)$ since $p^\star$ is a feasible solution to Problem (8.18) and p' is optimal, while $\Psi(p') \leq \Psi(p^\star)$ since $p^\star$ is an optimal δ-robust contract by definition. Then, it must be the case that $\Psi(p') = \Psi(p^\star)$.

Now, to complete the proof, consider any feasible solution $p \in \mathbb{R}^m_+$ to Problem (8.18). We show that $u^{\mathsf{P}}(p,a^\delta) \leq \Psi(p)$. Pick an arbitrary best-response action $a' \in A(p)$ and consider any $a \in A^\delta(p)$. By definition,

$$\begin{aligned} u^{\mathsf{A}}(p,a) &> u^{\mathsf{A}}(p,a') - \delta \\ &= \max_{a\in\mathcal{A}} u^{\mathsf{A}}(p,a) - \delta \\ &\geq u^{\mathsf{A}}(p,a^\star) - \delta. \end{aligned}$$

Hence, for Equation (8.18a) to hold for action a, it must be that $u^{\mathsf{P}}(p,a) \geq u^{\mathsf{P}}(p,a^\delta)$. Since the choice of a is arbitrary, this holds for every $a \in A^\delta(p)$. Consequently,

$$\Psi(p) = \min_{a\in A^\delta(p)} u^{\mathsf{P}}(p,a) \geq u^{\mathsf{P}}(p,a^\delta). \qquad \square$$

The above lemma implies that we can effectively "guess" $a^\star$ and a^δ, fixing these actions in Problem (8.18) and solving the optimization problem to obtain $p^\star$ (or

possibly a different, but still optimal, δ-robust contract). Since there are only $O(n^2)$ possible combinations of the values of $a^\star$ and a^δ, the approach is efficient as long as Problem (8.18) can be solved efficiently. A correct guess yields a contract p such that $\Psi(p) = \Psi(p^\star)$, whereas $\Psi(p) \leq \Psi(p^\star)$ for incorrect guesses. Thus, by comparing the Ψ values, we can identify a correct guess and a corresponding optimal δ-robust contract. It remains to show how to efficiently solve Problem (8.18).

Solving Problem (8.18). Problem (8.18) does *not* directly admit any efficient solution algorithm due to the non-convex constraint in Equation (8.18a). To deal with this issue, we rewrite Equation (8.18a) as follows:

$$\Big(\sum_{\omega\in\Omega} F_{a,\omega}p_\omega \leq c_a + u^{\mathsf{A}}(p,a^\star) - \delta\Big) \vee \Big(\sum_{\omega\in\Omega} F_{a,\omega}p_\omega \leq \sum_{\omega\in\Omega} F_{a,\omega}r_\omega - u^{\mathsf{P}}(p,a^\delta)\Big), \tag{8.19}$$

by expanding $u^{\mathsf{A}}(p,a) = \sum_{\omega\in\Omega} F_{a,\omega}p_\omega - c_a$ and $u^{\mathsf{P}}(p,a) = \sum_{\omega\in\Omega} F_{a,\omega}(r_\omega - p_\omega)$ and rearranging the terms. Hence, Equation (8.18a) is satisfied for action $a \in \mathcal{A}$ if and only if $\sum_{\omega\in\Omega} F_{a,\omega}p_\omega$ is smaller than the maximum of the right-hand sides of the two inequalities in Equation (8.19). The constraint effectively reduces to the second inequality for all $p \in \mathbb{R}^m_+$ such that

$$c_a + u^{\mathsf{A}}(p,a^\star) - \delta \leq \sum_{\omega\in\Omega} F_{a,\omega}r_\omega - u^{\mathsf{P}}(p,a^\delta), \tag{8.20}$$

while for the other $p \in \mathbb{R}^m_+$, it reduces to the first inequality.

Consequently, we can partition the contract space based on the satisfiability of Equation (8.20), considering all $a \in \mathcal{A}$. Within each subspace in the partition, only one inequality in Equation (8.19) is active for every $a \in \mathcal{A}$. So, effectively, Equation (8.19) reduces to a linear constraint for each action a, and the optimization problem to an LP. It then suffices to solve an LP for every subspace, each generating an optimal contract within its corresponding subspace. Among these contracts, the one providing the highest utility is an optimal solution to Problem (8.18).

Now, if the linear inequalities in Equation (8.20) (one for each action $a \in \mathcal{A}$) were n arbitrary inequalities, the above partition may consist of exponentially many subspaces, making the approach inefficient. Fortunately, the partition is much more well-structured, since the hyperplanes corresponding to the inequalities are *parallel*, as the coefficients of p in Equation (8.20) are invariant with respect to a. As a result, they partition the space into only $O(n)$ subspaces.

Next, we show how to exploit the above observation to solve Problem (8.18), by solving $O(n)$ suitable subproblems instead.

Formulating the Subproblems. Let us first rearrange Equation (8.20) as follows:

$$u^{\mathsf{A}}(p,a^\star) + u^{\mathsf{P}}(p,a^\delta) - \delta \leq \nu_a \coloneqq \sum_{\omega\in\Omega} F_{a,\omega}r_\omega - c_a, \tag{8.21}$$

where ν_a is exactly the social welfare generated by action a (which is independent of the specific contract adopted). Then, we can re-order agent's actions $a_1, \ldots, a_n$ in such a way that $\nu_{a_1} \leq \nu_{a_2} \leq \cdots \leq \nu_{a_n}$. For simplicity, we write $\nu_j = \nu_{a_j}$, and we let $\nu_0 = -\infty$ and $\nu_{n+1} = +\infty$. Then, the following lemma is straightforward.

Lemma 8.3. *For every contract $p \in \mathbb{R}^m_+$, if it holds that*

$$\nu_{j-1} \leq u^{\mathsf{A}}(p, a^\star) + u^{\mathsf{P}}(p, a^\delta) - \delta \leq \nu_j,$$

then:

- $\sum_{\omega\in\Omega} F_{a,\omega} p_\omega \leq \sum_{\omega\in\Omega} F_{a,\omega} r_\omega - u^{\mathsf{P}}(p, a^\delta) \iff$ *Equation* (8.19) *holds for all actions* $a \in \{a_\ell \mid \ell \leq j-1\}$*; and*
- $\sum_{\omega\in\Omega} F_{a,\omega} p_\omega \leq c_a + u^{\mathsf{A}}(p, a^\star) - \delta \iff$ *Equation* (8.19) *holds for all actions* $a \in \{a_\ell \mid j \leq \ell\}$.

In other words, the condition in the lemma defines a suitable subspace of contracts $\mathcal{P}_j$ for each $j \in \{1, \ldots, n+1\}$. The following LP solves for a $p \in \mathbb{R}^m_+$ that is optimal within $\mathcal{P}_j$.

$$\max_{p\in\mathbb{R}^m_+} \quad u^{\mathsf{P}}(p, a^\delta) \tag{8.22}$$

subject to the following constraints:

$$\nu_{j-1} \leq u^{\mathsf{A}}(p, a^\star) + u^{\mathsf{P}}(p, a^\delta) - \delta \leq \nu_j \tag{8.22a}$$

$$\sum_{\omega\in\Omega} F_{a,\omega} p_\omega \leq \sum_{\omega\in\Omega} F_{a,\omega} r_\omega - u^{\mathsf{P}}(p, a^\delta) \quad \forall a \in \{a_\ell \mid \ell \leq j-1\} \tag{8.22b}$$

$$\sum_{\omega\in\Omega} F_{a,\omega} p_\omega \leq c_a + u^{\mathsf{A}}(p, a^\star) - \delta \qquad \forall a \in \{a_\ell \mid j \leq \ell\}. \tag{8.22c}$$

Since $\bigcup_{j=1}^{n+1} \mathcal{P}_j = \mathbb{R}^m_+$, solving the LP in Problem (8.22) for all $j \in \{1, \ldots, n+1\}$ and picking the best among the obtained solutions gives an optimal solution to Problem (8.18).

Remark 8.2. *The left-hand side of Equation* (8.21) *is roughly (up to a δ difference) the social welfare of an optimal contract. Thus, Equation* (8.3) *can be interpreted as follows. For low-social-welfare actions $a_1, \ldots, a_{j-1}$, yielding a sufficiently high utility for the principal automatically provides a low utility for the agent. This fulfils the first inequality in Equation* (8.19) *(and Equation* (8.18a)*). Conversely, for high-social-welfare actions $a_j, \ldots, a_n$, a sufficiently low utility for the agent automatically gives a high utility for the principal. This fulfils the second inequality in Equation* (8.19) *(and Equation* (8.18a)*).*

Putting All Together. We summarize our results in this section into Equation (8.2) and the following main theorem.

Theorem 8.2. *Equation* (8.2) *computes an optimal δ-robust contract in polynomial time.*

Proof Sketch. The polynomial runtime of Equation (8.2) is obvious as it enumerates $O(n^3)$ value combinations and solves an LP for each of them. To see the correctness of the algorithm, note that the inner for-loop of Equation (8.2) effectively solves Problem (8.18), for the pair $(a^\star, a^\delta)$ enumerated in the outer for-loop. Now, if $a^\star$ and a^δ happen to be the agent's exact and δ-best responses, respectively, under some optimal contract, then according to Equation (8.2), the outer loop produces an optimal contract p', with $\Psi(p') \geq \Psi(p)$ for all $p \in \mathbb{R}^m_+$. By

comparing the Ψ values, the algorithm identifies and outputs such an optimal contract. □

Algorithm 8.2 Compute an optimal δ-robust contract

$p^\star \leftarrow$ `null`, $\psi^\star \leftarrow -\infty$
for $(a^\star, a^\delta) \in \mathcal{A} \times \mathcal{A}$ **do**
  for $j = 1, \dots, n+1$ **do**
    Solve Problem (8.22) instantiated with $(a^\star, a^\delta)$ s.t. Equation (8.22a)-(8.22c) are satisfied, and let an optimal solution be p'
    if $\Psi(p') > \psi^\star$ **then**
      $p^\star \leftarrow p'$
      $\psi^\star \leftarrow \Psi(p')$
    end if
  end for
end for
return $p^\star$

8.5 Related works

Two works that are closely related to this chapter are [Gan et al., 2023a, Yang and Zhang, 2025], which consider *robustness* notions analogous to ours, though in different settings. Specifically, Gan et al. [2023a] initiate this research line, by studying the problem of computing robust leader's commitments in *Stackelberg games* where the follower plays an approximate best response. They show that it is NP-hard to approximate an optimal robust commitment of the leader and, in accordance to this hardness result, they provide a quasi-polynomial-time approximation scheme (QPTAS). Yang and Zhang [2025] extend the study initiated by Gan et al. [2023a] to *Bayesian persuasion*, with the goal of computing robust signaling schemes under approximate best responses of the receiver. Similarly to Gan et al. [2023a], they show that computing an approximately-optimal robust signaling scheme is NP-hard and provide a QPTAS. In sharp contrast, we show that, in hidden-action principal-agent problems, an optimal robust contract can be computed efficiently.

Our chapter is also related to other research lines addressing different concepts of *robustness* in contract design. For instance, Carroll [2015] studies settings where the principal only knows a superset of agents' actions, while Dütting et al. [2019] introduce a different notion of uncertainty in which the principal has partial knowledge of the distributions over outcomes associated with agent's actions. Both these works show that linear contracts are a sufficient class of contracts to determine the *min-max robust* optimal contract. Notice that these two frameworks differ from ours, as within our framework, when the robustness parameter $\delta \in (0, 1)$ is arbitrarily small, the problem becomes very close to the classical version of the hidden-action principal-agent problem, in which it is known that linear contracts are *not* generally optimal (see, *e.g.*, [Dütting et al., 2019]).

CHAPTER 9

Conclusions and future research

In this thesis, we designed learning algorithms for the generalized principal-agent problem. These algorithms play a crucial role in today's digital economies, as they provide a framework for adaptive decision-making in settings where a central platform interacts with large populations of self-interested, rational users. A key advantage of the proposed approaches is that they do not require the principal to know the agents' features in advance, making them suitable for large-scale deployment in complex and dynamic environments.

As a future step, we intend to consider similar problems in a Bayesian setting, *i.e.*, when the agent's payoff-relevant information is sampled according to some unknown probability distribution, or cases in which the agent may be farsighted and not play a best-response action at each round. Indeed, in this work we focused on myopic agents that maximize their reward instantaneously. This assumption is well motivated by the fact that platforms, firms, and agencies often interact with a large number of users, and the same user may not appear repeatedly or may have little incentive to plan strategically across interactions. However, there could be situations in which the agent is aware of the long-term consequences of their actions and adjusts their behavior accordingly, leading to a dynamic strategic interaction that requires a more sophisticated analysis.

Another interesting research direction concerns the use of AI tools, such as large language models (LLMs), within this framework. For instance, LLMs could be employed to generate persuasive text messages to be sent to receivers, or to draft the content of contracts to be offered to agents. The integration of such models could enable the principal to design more effective communication strategies that adapt to the agent's characteristics and preferences, and to deploy AI agents that

account for the agent's strategic behavior and adjust their actions accordingly. The adoption of these agents opens the way to new challenges, both from an empirical and a theoretical point of view. Indeed, from a theoretical perspective, these settings require the design of new algorithmic approaches that either combine existing methods or introduce novel ones capable of capturing the interplay between learning, communication, and strategic behavior.

APPENDIX $\mathcal{A}$

Additional technical lemmas from Chapter 3

We now present two useful lemmas. Lemma A.1 provides a lower bound on the sum of rational numbers represented with at most $B > 0$ bits. Lemma A.2, instead, provides a bound on the bit-complexity of the vertices of the regions $\mathcal{U}_j$ computed during the execution of Algorithm 3.1.

Lemma A.1. *Given $q_1, \ldots, q_m \in \mathbb{Q}$ represented as fractions, each with bit-complexity bounded by $B \in \mathbb{N}_+$, the bit-complexity of their sum $q := \sum_{i=1}^{m} q_i$ is bounded by $4B$ when $m = 2$, while it is bounded by $3mB$ when $m > 2$. Moreover, the absolute value of their sum is either equal to 0 or it is greater than or equal to 2^{-Bm}.*

Proof. To prove the lemma, we define the sum of $q_1, \ldots, q_m$ as $q \in \mathbb{Q}$, and we express each q_i with $i \in [m]$ as $\frac{\alpha_i}{\beta_i}$, where both $\alpha_i, \beta_i \in \mathbb{Z}$ for each $i \in [m]$. Then the following holds:

$$q = \sum_{i=1}^{m} q_i = \sum_{i=1}^{m} \frac{\alpha_i}{\beta_i} = \frac{\displaystyle\sum_{i=1}^{m} \alpha_i \left(\prod_{i \neq j} \beta_j \right)}{\displaystyle\prod_{i \in [m]} \beta_i}.$$

It is easy to see that the bit complexity of the denominator that defines the rational number q is equal to $\sum_{i \in [m]} B_{\beta_i}$, while the absolute value of the numerator

of q can be upper-bounded as follows:

$$\left|\sum_{i=1}^{m}\alpha_i\prod_{i\neq j}\beta_j\right| \leq m\left|\left(\alpha_{i^*}\prod_{i\neq i^*}\beta_i\right)\right| \leq 2^{\log_2(m)+B_{\alpha_{i^*}}+\sum_{i\neq i^*}B_{\beta_i}},$$

where we let $i^* \in \arg\max_{i\in[m]}\alpha_i\prod_{i\neq j}\beta_j$. As a result, the number of bits required to encode q can be bounded as:

$$\begin{aligned}B_q &\leq \log_2(m) + B_{\alpha_{i^*}} + \sum_{i\neq i^*}B_{\beta_i} + \sum_{i\in[m]}B_{\beta_i}\\ &\leq \log_2(m) + B + 2\sum_{i\neq i^*}B_{\beta_i}\\ &\leq \log_2(m) + (2m-1)B \leq 3mB.\end{aligned}$$

The second inequality above holds because $B_{\alpha_i}+B_{\beta_i}\leq B$, as the bit complexity of each q_i with $i\in[m]$ is bounded by B. Moreover, when $m=2$, we have that $B_q \leq \log_2(2)+(4-1)B \leq 4B$. Finally, we observe that the absolute value of q is either zero or can be lower bounded as shown in the following:

$$\left|\sum_{i=1}^{m}q_i\right| = \left|\sum_{i=1}^{m}\frac{\alpha_i}{\beta_i}\right| = \left|\frac{\sum_{i=1}^{m}\alpha_i\left(\prod_{i\neq j}\beta_j\right)}{\prod_{i\in[m]}\beta_i}\right| \geq \frac{1}{2^{mB}},$$

which concludes the proof. □

Lemma A.2. *Given an SG, if follower's payoffs are represented as fractions with bit-complexity bounded by L, then each vertex $v\in V(\mathcal{U}_j)$ of an $\mathcal{U}_j$ computed by Algorithm 3.1 has bit-complexity at most $9Lm^2$. Furthermore, with a bit-complexity of $9Lm^2$, all the components of the vector identifying a vertex can be written as fractions with the same denominator.*

Proof. Let v be a vertex belonging to an upper bound $\mathcal{U}_j$ such that $\text{vol}(\mathcal{U}_j) > 0$. Then such a vertex lies on the hyperplane Δ_m^+ ensuring that the sum of its components is equal to one. Furthermore, it also belongs to a subset of $m-1$ linearly independent hyperplanes, which can pertain either to the boundary planes denoted as H_i with $i\in[m]$, or to the separating hyperplanes between two followers' best response regions, *i.e.*, H_{jk} with $a_j, a_k \in \mathcal{A}_f$. Consequently, there exists a matrix $A\in\mathbb{Q}^{m\times m}$ and a vector $b\in\mathbb{Q}^m$ such that $Av=b$. The entries of the matrix A may encode the difference in terms of follower's utility between two follower actions a_j and a_k in $\mathcal{A}_f$ for each leader's action $a_i\in\mathcal{A}_\ell$, *i.e.*, $u_f(a_i,a_j)-u_f(a_i,a_k)$. As a result, the bit complexity of each entry of the matrix A can be bounded by $4L$, as a consequence of Lemma A.1, and by observing

that the coefficients that defines the hyperplanes Δ_m^+ or H_i, with $i \in [m]$, are either 0 or 1. Therefore, we can multiply each row of the matrix $A_i \in \mathbb{Q}^m$ and the corresponding $b_i \in \mathbb{Q}$ with $i \in [m]$, by a constant bounded by 2^{4Lm}.

In this way, we can formulate the previous linear systems to compute the vertex v using a matrix $A' \in \mathbb{Z}^{m \times m}$ and a vector $b' \in \mathbb{Z}^m$, such that $A'v = b'$, where each component of A' and b' satisfies $|a'_{ij}| \leq 2^{4Lm}$ and $|b'_j| \leq 2^{4Lm}$ for each $i, j \in [m]$. To prove the lemma, we employ a similar approach to the one presented in Lemma 8.2 by Bertsimas and Tsitsiklis [1997], and we define $A'(j)$ as the matrix obtained by substituting the j-th column of A' with b'. Then, by Cramer's rule, the value of the j-th component of v_j can be computed as follows:

$$v_j = \frac{\det(A'(j))}{\det(A')} \quad \forall j \in [m].$$

We observe that both determinants are integer numbers as the entries of both A' and b' are all integers, thus by Hadamard's inequality we have:

$$\begin{aligned}
|\det(A')| &\leq \prod_{i \in [m]} \sqrt{\sum_{j \in [m]} {a'_{ji}}^2} \\
&\leq \prod_{i \in [m]} \sqrt{\sum_{j \in [m]} (2^{4Lm})^2} \\
&= \prod_{i \in [m]} \sqrt{m(2^{4Lm})^2} \\
&= (2^{4Lm^2}) m^{\frac{m}{2}}.
\end{aligned}$$

Furthermore, observing that the same upper bound holds for $|\det(A^j)|$, the bit complexity of v is bounded by:

$$\begin{aligned}
2\left\lceil \log_2(2^{4Lm^2} m^{m/2}) \right\rceil &\leq 2(\log_2(2^{4Lm^2} m^{m/2}) + 1) \\
&= 8Lm^2 + m\log_2(m) + 2 \\
&\leq 9Lm^2.
\end{aligned}$$

Finally, we notice that this upper bound holds when every component of the vertex has the same denominator $\det(A')$, concluding the proof. □

APPENDIX $\mathcal{B}$

Additional technical lemmas from Chapter 4

In this section we present the proof of Lemma 4.3 and the different procedures used during the execution of Algorithm 4.3 . As a first step, for the sake of analysis, we introduce the definition of a *clean event* under which the `Action-Oracle` procedure always returns the action $a(x) \in \mathcal{A}$, as formally stated below.

Definition B.1 (Clean event of `Action-Oracle`)**.** *We denote $\mathcal{E}^{\mathrm{a}}$ as the event in which Algorithm 4.3 correctly returns the follower's best response $a(x)$ whenever executed.*

We now discuss the proof of Lemma 4.3. Intuitively, to prove the lemma we observe that, thanks to Lemma 4.2 and the definition of $\mathcal{X}_\epsilon$, it is possible to bound the number of rounds required by Algorithm 4.3 to ensure that it returns the best response $a(x) \in \mathcal{A}$ with high probability in a number of rounds proportional to $1/\epsilon$.

Lemma 4.3. *Given inputs $\mathcal{X}_\epsilon \subseteq \mathcal{X}$ and $\zeta \in (0,1)$ for Algorithm 4.4, let $L := B + B_\epsilon + B_{\widehat{\mu}}$, where B_ϵ, and $B_{\widehat{\mu}}$ denote the bit-complexity of ϵ, and $\widehat{\mu}$, respectively. Then, under event $\mathcal{E}_1$ and with at probability at least $1-\zeta$, Algorithm 4.4 outputs a $\mathcal{Y} := (\mathcal{V}, (a(x))_{x\in\mathcal{V}})$ such that*

$$\mathcal{V} = \bigcup_{a_i \in \mathcal{A}:\mathrm{vol}(\mathcal{X}_\epsilon(a_i))>0} V(\mathcal{X}_\epsilon(a_i))$$

in a number of rounds T_2 satisfying:

$$T_2 \leq \widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\zeta}\right)\left(d^7 L + \binom{d+n}{d}\right)\right).$$

Proof. We denote with $L = B + B_\epsilon + B_{\widehat{\mu}}$. Thanks to Lemma B.1 and Lemma B.2, with an approach similar to the one proposed in Theorem 3.1 in Chapter 3, we can prove that, under the event $\mathcal{E}^{\text{a}}$, with probability at least $1-\delta n^2(5(d+n)^2+n)$, Algorithm 4.4 computes every polytope $\mathcal{X}_\epsilon(a)$ with volume larger than zero by performing at most:

$$\mathcal{O}\left(n^2\left(d^7 L \log(1/\delta) + \binom{d+n}{d}\right)\right),$$

calls to the `Action-Oracle` procedure.

Furthermore, let $\zeta = 2\delta n^2(5(d+n)+n)$ be the value defined at Line 1 in Algorithm 4.4, and let $C \in \mathbb{N}$ denote the number of calls performed by Algorithm 4.4 to the `Action-Oracle` procedure. Thus, under the event $\mathcal{E}^{\text{a}}$, with probability at least $1 - \zeta/2$, we have

$$C \leq \widetilde{\mathcal{O}}\left(n^2\left(d^7 L \log(1/\zeta) + \binom{d+n}{d}\right)\right).$$

Moreover, by setting $\rho = \zeta/2C$, and by Lemma 4.2, with probability at least $1-\rho$, every execution of `Action-Oracle` requires at most $N \in \mathbb{N}$ rounds, where N can be bounded as follows:

$$N \leq \mathcal{O}\left(\frac{\log(1/\rho)}{\epsilon}\right) = \mathcal{O}\left(\frac{1}{\epsilon}\log\left(\frac{2C}{\zeta}\right)\right). \tag{B.1}$$

Consequently, since the number of calls to the `Action-Oracle` procedure is equal to C, by employing a union bound, the probability that each one of these calls requires at most N rounds to terminate is greater than or equal to:

$$1 - C\rho = 1 - \frac{\zeta}{2C}C = 1 - \frac{\zeta}{2}.$$

To conclude the proof, we employ a union bound over the probability that every execution of `Action-Oracle` terminates in at most N rounds, and the probability that Algorithm 4.4 performs C calls to the `Action-Oracle` procedure. Since the probability that these two events hold is of at least $1 - \zeta/2$, with probability at least $1 - \zeta$, Algorithm 4.4 correctly terminates by using a number of samples of the order:

$$\widetilde{\mathcal{O}}\left(\frac{n^2}{\epsilon}\log^2\left(\frac{1}{\zeta}\right)\left(d^7 L + \binom{d+n}{d}\right)\right),$$

concluding the proof. □

B.0.1 The `Find-HS` procedure

The goal of the `Find-HS` procedure is to compute a new separating hyperplane H_{jk} between a given region $\mathcal{X}_\epsilon(a_j)$ and some other polytope $\mathcal{X}_\epsilon(a_k)$, where a_j and a_k belong to $\mathcal{A}$. To do so, it receives as input an upper bound $\mathcal{U}(a_j)$ of some polytope $\mathcal{X}_\epsilon(a_j)$, an interior point $x^{\text{int}} \in \text{int}(\mathcal{U}(a_j))$, a vertex $v \in \text{V}(\mathcal{U}(a_j))$ that does not

belong to $\mathcal{X}_\epsilon(a_j)$, and a parameter $\delta > 0$ as required by the `Sample-Int` procedure. As a first step, Algorithm B.1 samples at random a slice x from the interior of the upper bound $\mathcal{U}(a_j)$. Subsequently, it performs a binary search on the segment between x and either v or x^{int}, depending on the best response $a(x)$ in x. This binary search returns a point x° on some new separating hyperplane H_{jk} (Line 8).

As a further step, the algorithm computes two sets of normalized slices, $\mathcal{S}_j \subseteq \mathcal{X}_\epsilon(a_j)$ and $\mathcal{S}_k \subseteq \mathcal{X}_\epsilon(a_k)$. Finally, it performs $d-1$ binary searches between different couples of points, one in $\mathcal{S}_j$ and the other in $\mathcal{S}_k$, in order to completely identify the separating hyperplane.

Algorithm B.1 `Find-HS`

Require: $a_j, \mathcal{U}(a_j), x^{\text{int}}, v, \delta$
1: $x \leftarrow$ `Sample-Int`$(\mathcal{U}(a_j), \delta)$
2: $x^1 \leftarrow x$
3: **if** `Action-Oracle`$(x) = a_j$ **then**
4: $\quad x^2 \leftarrow v$
5: **else**
6: $\quad x^2 \leftarrow x^{\text{int}}$
7: **end if**
8: $x^\circ \leftarrow$ `Binary-Search`(a_j, x^1, x^2)
9: $\alpha \leftarrow 2^{-4d(B_x+B+B_{\widehat{\mu}}+B_\epsilon)}/d$
10: $\mathcal{S}_j \leftarrow \varnothing; \mathcal{S}_k \leftarrow \varnothing$
11: **for** $i = 1 \dots d$ **do**
12: $\quad x \leftarrow$ `Sample-Int`$(H_i \cap \mathcal{X}, \delta)$
13: $\quad x^{+i} \leftarrow x^\circ + \alpha(x - x^\circ)$
14: $\quad x^{-i} \leftarrow x^\circ - \alpha(x - x^\circ)$
15: $\quad$ **if** `Action-Oracle`$(x^{+i}) = a_j$ **then**
16: $\quad\quad \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{x^{+i}\} \wedge \mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{x^{-i}\}$
17: $\quad$ **else**
18: $\quad\quad \mathcal{S}_k \leftarrow \mathcal{S}_k \cup \{x^{+i}\} \wedge \mathcal{S}_j \leftarrow \mathcal{S}_j \cup \{x^{-i}\}$
19: $\quad$ **end if**
20: **end for**
21: Build H_{jk} by `Binary-Search`(a_j, x^1, x^2) for $d-1$ pairs of linearly-independent points $x^1 \in \mathcal{S}_j, x^2 \in \mathcal{S}_k$

Lemma B.1. *With probability at least $1 - 5(d+n)^2\delta$, under the event $\mathcal{E}^{\text{a}}$, Algorithm B.1 returns a separating hyperplane H_{jk} by using $\mathcal{O}(d^7(B + B_\epsilon + B_{\widehat{\mu}}) + d^4 \log(1/\delta))$ calls to Algorithm 6.2.*

Proof. We observe that, with the same analysis provided in Lemma 3.7, we can prove that, under the event $\mathcal{E}^{\text{a}}$, the binary-search procedure described in Algorithm B.2 correctly computes a point on a separating hyperplane by calling the `Action-Oracle` procedure at most $\mathcal{O}(d(B_x + B))$ times.

With the same reasoning employed in Lemma 3.4 and 3.5, we can prove that with probability at least $1 - 5(d+n)^2\delta$, under the event $\mathcal{E}^{\text{a}}$, the points x^{+i} are linearly independent and do not belong to H_{jk}. To conclude the proof, we have

to show that every x^{+i} belongs to either $\mathcal{X}_\epsilon(a_j)$ or $\mathcal{X}_\epsilon(a_k)$. This is because, if the previous condition holds, under the event $\mathcal{E}^{\text{a}}$, Algorithm B.1 correctly computes a new separating hyperplane with probability at least $1 - 5(d+n)^2\delta$.

To do that, we show that the constant α defined at Line 9 in Algorithm B.1 is such that all the points x^{+i} and x^{-i} either belong to $\mathcal{X}_\epsilon(a_j)$ or $\mathcal{X}_\epsilon(a_k)$, given that x° belongs to the hyperplane between these polytopes. With an argument similar to the one proposed in Chapter 3, the distance between x^i and any separating hyperplane can be lower bounded by $2^{-d(B_x+4B)}$, where B_x is the bit-complexity of x°. Similarly, the distance between x° and the hyperplane

$$\widehat{H} = \{x \in \mathbb{R}^d \mid \sum_{\theta \in \widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\}$$

can be lower bounded as follows:

$$d(x^\circ, \widehat{H}) = \frac{\left|\sum_{\theta\in\widetilde{\Theta}} x_\theta \widehat{\mu}_\theta + 2\epsilon\right|}{\sqrt{\sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}_\theta^2}} \geq \frac{1}{d2^{3d(B_x+B_{\widehat{\mu}}+B_\epsilon)}}, \tag{B.2}$$

where $B_{\widehat{\mu}}$ is the bit-complexity of $\widehat{\mu}$. The inequality follows by observing that the denominator of the fraction above is at most d, while to lower bound the numerator we define the following quantities:

$$\sum_{\theta\in\widetilde{\Theta}} x_\theta \widehat{\mu}_\theta = \frac{\alpha}{\beta} \quad \text{and} \quad \epsilon = \frac{\gamma}{\nu},$$

where α and β are integers numbers, while γ and ν are natural numbers. Thus, the numerator $\left|\sum_{\theta\in\widetilde{\Theta}} x_\theta \widehat{\mu}_\theta + 2\epsilon\right|$ of the fraction defined in Equation (B.2) can be lower bounded as follows:

$$\left|\sum_{\theta\in\widetilde{\Theta}} x_\theta \widehat{\mu}_\theta + 2\epsilon\right| = \left|\frac{\alpha\nu + 2\beta\gamma}{\beta\nu}\right| \geq \left|\frac{1}{\beta\nu}\right| \geq 2^{-3d(B_x+B_{\widehat{\mu}}+B_\epsilon)},$$

where the last inequality follows from the fact that the bit-complexity of ν is at most B_ϵ, while the bit-complexity of β cannot exceed $3d(B_x + B_{\widehat{\mu}})$ as stated by Lemma A.1.

Overall, the distance between x° and the boundary of the $\mathcal{X}_\epsilon(a_j) \cap \mathcal{X}_\epsilon(a_k)$ is strictly larger than α. Thus, every slice x^{+i} and x^{-i} belongs to $\mathcal{X}_\epsilon$ and either $\mathcal{X}(a_j)$ or $\mathcal{X}(a_k)$. Finally, we observe that the bit-complexity of x is such that

$$B_x = \mathcal{O}(d^3(B + B_\epsilon + B_{\widehat{\mu}}) + \log(1/\delta)),$$

as stated by Lemma B.2. Thus, the first binary-search requires

$$\mathcal{O}(d(B_x + B)) = \mathcal{O}(d^4 L + d\log(1/\delta))$$

calls to `Action-Oracle`, where $L = B + B_\epsilon + B_{\widehat{\mu}}$.

Furthermore, the bit-complexity of x° is bounded by $\mathcal{O}(d^4 L + d\log(1/\delta))$. As a result, the bit-complexity of the slices x^{+i} and x^{-i} is $\mathcal{O}(d^5 L + d\log(1/\delta))$, given

that the bit-complexity of α is $\mathcal{O}(dL)$. It follows that each binary search between two points in $\mathcal{S}_j$ and $\mathcal{S}_k$ requires $\mathcal{O}(d^6 L + d^2 \log(1/\delta))$ calls to `Action-Oracle`.

Overall, Algorithm B.1 invokes the `Action-Oracle` procedure at most

$$\mathcal{O}(d^7 L + d^4 \log(1/\delta))$$

times, accounting for the $d-1$ binary searches, concluding the proof. □

B.0.2 The `Binary-Search` procedure

Algorithm B.2 `Binary-Search`

Require: a_j, x^1, x^2 of bit-complexity bounded by some $B_x > 0$

$\lambda_1 \leftarrow 0$
$\lambda_2 \leftarrow 1$
while $|\lambda_2 - \lambda_1| \geq 2^{-6d(5B_x+8B)}$ **do**
  $\lambda \leftarrow (\lambda_1 + \lambda_2)/2$; $x^\circ \leftarrow x^1 + \lambda(x^2 - x^1)$
  if `Action-Oracle`$(x^\circ) = a_j$ **then**
    $\lambda_1 \leftarrow \lambda$
  else
    $\lambda_2 \leftarrow \lambda$
  end if
end while
$\lambda \leftarrow$ `Stern-Brocot-Tree`$(\lambda_1, \lambda_2, 3d(5B_x + 8B))$
$x^\circ \leftarrow \lambda x^1 + (1-\lambda)x^2$

The `Binary-Search` procedure performs a binary search on the segment connecting two points, $x^1, x^2 \in \mathcal{X}_\epsilon$ such that $x^1 \in \mathcal{X}_\epsilon(a_j)$ and $x^2 \notin \mathcal{X}_\epsilon(a_j)$ for some $a_j \in \mathcal{A}$, in order to find a point x° on some separating hyperplane H_{jk}. At each iteration, the binary search queries the middle point of the segment. Depending on the receiver's best-response in such a point, it keeps one of the two halves of the segment for the subsequent iteration.

The binary search ends when the segment is sufficiently small, so that it contains a single point with a bit-complexity appropriate for a point that lies on both the hyperplane H_{jk} and the segment connecting x^1 and x^2. Such a point can be found traversing the Stern-Brocot-Tree. For an efficient implementation, see [Forišek, 2007]. Overall, Algorithm B.2 performs $\mathcal{O}(d(B_x+B))$ calls to the `Action-Oracle` procedure, and returns a normalized slice x° with bit-complexity bounded by $\mathcal{O}(d(B_x + B))$, where B_x is the bit-complexity of the points x^1 and x^2.

B.0.3 The `Sample-Int` procedure

The `Sample-Int` procedure (Algorithm B.3) samples at random a normalized slice from the interior of a given polytope $\mathcal{P}$. We observe that each polytope Algorithm 4.4 is required to sample from is defined as the intersection of $\mathcal{X}_\epsilon$ with some separating half-spaces as the ones defined in Section 4.2. This procedure provides theoretical guarantees both on the bit-complexity of the point x being sampled and on the probability that such a point belongs to a given hyperplane.

Furthermore, it can be easily modified to sample a point from a facet of the simplex $\mathcal{X} = \Delta_d$ (intuitively, this is equivalent to sample a point from Δ_{d-1}). As a first step, Algorithm B.3 computes a normalized slice $x^\diamond$ in the interior of $\mathcal{P}$ (Line 2). Subsequently, it samples randomly a vector y from a suitable grid belonging to the $(d-1)$-dimensional hypercube with edges of length 2. As a further step, it sums each component $x_i^\diamond$ of the normalized slice $x^\diamond$ with the corresponding component y_i of y, scaled by an opportune factor ρ, where the constant ρ is defined to ensure that x belongs to the interior of $\mathcal{P}$.

Algorithm B.3 `Sample-Int`

Require: $\mathcal{P} \subseteq \mathcal{X}_\epsilon : \text{vol}_{d-1}(\mathcal{P}) > 0$, and δ
1: $\mathcal{V} \leftarrow d$ linearly-independent vertexes of $\mathcal{P}$
2: $x^\diamond \leftarrow \frac{1}{d}\sum_{v\in\mathcal{V}} v$
3: $\rho \leftarrow \left(d^3 2^{9d^3L+4dL}\right)^{-1}$; $M \leftarrow \lceil \sqrt{d}/\delta \rceil$
4: $y \sim \text{Uniform}(\{-1, -\frac{M-1}{M}, \ldots, 0, \ldots, \frac{M-1}{M}, 1\}^{d-1})$
5: **for all** $i \in 1, \ldots, d-1$ **do**
6: $\quad x_i \leftarrow x_i^\diamond + \rho y_i$
7: **end for**
8: $x_d \leftarrow 1 - \sum_{i=1}^{d-1} x_i$

The theoretical guarantees provided by Algorithm B.3 are formalized in the following lemma:

Lemma B.2. *Given a polytope $\mathcal{P} \subseteq \mathcal{X}_\epsilon : \text{vol}_{d-1}(\mathcal{P}) > 0$ defined by separating or boundary hyperplanes, Algorithm B.3 computes $x \in \text{int}(\mathcal{P})$ such that, for every linear space $H \subset \mathbb{R}^d : \mathcal{P} \not\subseteq H$ of dimension at most $d-1$, the probability that $x \in H$ is at most δ. Furthermore, the bit-complexity of x is $\mathcal{O}(d^3(B + B_\epsilon + B_{\widehat{\mu}}) + \log(1/\delta))$.*

Proof. In the following, we prove that x belongs to the interior of the polytope $\mathcal{P}$. To do that, we observe that the point $x^\diamond$ belongs to the interior of $\mathcal{P}$, while, with the same analysis proposed in Lemma 3.8, the distance between $x^\diamond$ and x can be upper bounded by ρn. To ensure that x belongs to $\text{int}(\mathcal{P})$, we have to show that $d(x^\diamond, x)$ is smaller than the distance between $x^\diamond$ and any hyperplane defining the boundary of $\mathcal{P}$.

Let us denote with v^h the h-vertex of the set $\mathcal{V}$, so that $\mathcal{V} = \{v^1, v^2, \ldots, v^d\}$. Furthermore, we define:

$$v_\theta^h := \frac{\gamma_\theta^h}{\nu_h}$$

for each $h \in [d]$ and $\theta \in \Theta$. Since $x^\diamond$ belongs to $\text{int}(\mathcal{P})$, the distance between $x^\diamond$ and any separating hyperplane H_{jk} can be lower bounded as follows:

$$d(x^\diamond, H_{jk}) = \left| \frac{\sum_{\theta\in\Theta} x_\theta^\diamond \mu_\theta (u_\theta(a_j) - u_\theta(a_k))}{\sqrt{\sum_{\theta\in\Theta} \mu_\theta^2 (u_\theta(a_j) - u_\theta(a_k))^2}} \right|$$

$$= \left| \frac{\sum_{\theta\in\Theta}\sum_{h=1}^{d} v_\theta^h \mu_\theta(u_\theta(a_j) - u_\theta(a_k))}{d\sqrt{\sum_{\theta\in\Theta}\mu_\theta^2(u_\theta(a_j) - u_\theta(a_k))^2}} \right|$$
$$\geq \frac{1}{d^2} 2^{-4dB - 9d^3(B + B_\epsilon + B_{\widehat{\mu}})}.$$

To prove the last inequality, we observe that the denominator of the fraction above can be upper bounded by d^2. To lower bound the numerator, we define:

$$\mu_\theta(u_\theta(a_j) - u_\theta(a_k)) \coloneqq \frac{\alpha_\theta}{\beta_\theta}$$

for each $\theta \in \Theta$. As a result, we have:

$$\left| \sum_{\theta\in\Theta}\sum_{h=1}^{d} v_\theta^h \mu_\theta(u_\theta(a_j) - u_\theta(a_k)) \right| = \left| \sum_{\theta\in\Theta}\sum_{h=1}^{d} \frac{\alpha_\theta \gamma_\theta^h}{\beta_\theta \nu_h} \right|$$
$$= \left| \frac{\sum_{\theta\in\Theta}\sum_{h=1}^{d} \alpha_\theta \gamma_\theta^h \left(\prod_{\theta'\neq\theta} \beta_{\theta'} \prod_{h'\neq h} \nu_{h'}\right)}{\prod_{\theta\in\Theta}\beta_\theta \prod_{h=1}^{d} \nu_h} \right|$$
$$\geq \left(\prod_{\theta\in\Theta} \beta_\theta \prod_{h=1}^{d} \nu_h \right)^{-1}$$
$$\geq 2^{-4dB - 9d^3(B + B_\epsilon + B_{\widehat{\mu}})}.$$

The first inequality holds because the numerator of the fraction above can be lower bounded by one. The denominator can be instead upper bounded observing that the bit-complexity of each β_θ is at most $4B$ while the bit-complexity of each ν_h is at most $9d^2(B + B_\epsilon + B_{\widehat{\mu}})$, as stated by Lemma B.3. In a similar way, we can lower bound the distance between $x^\diamond$ and

$$\widehat{H} \coloneqq \{x \in \mathbb{R}^d \mid \sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\},$$

as follows:

$$d(x^\diamond, \widehat{H}) = \left| \frac{\sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}_\theta x_\theta^\diamond + 2\epsilon}{\sqrt{\sum_{\theta\in\widetilde{\Theta}} \widehat{\mu}^2}} \right|$$
$$= \left| \frac{\sum_{\theta\in\widetilde{\Theta}}\sum_{h=1}^{d} \widehat{\mu}_\theta v_\theta^h + 2\epsilon}{d\sqrt{\sum_{\theta\in\widetilde{\Theta}} \hat{\mu}^2}} \right|$$
$$\geq \frac{1}{d^2} 2^{-B_{\widehat{\mu}} - B_\epsilon - 9d^3(B + B_\epsilon + B_{\widehat{\mu}})}.$$

To prove the last inequality, we observe that the denominator of the fraction above can be upper bounded by d^2, while to lower bound the numerator, we

define:

$$\hat{\mu}_\theta \coloneqq \frac{N_\theta}{p} \text{ and } \epsilon = \frac{\alpha}{\beta}$$

for each $\theta \in \widetilde{\Theta}$. As a result, we have:

$$\begin{aligned}
\left| \sum_{\theta \in \widetilde{\Theta}} \sum_{h=1}^{d} \widehat{\mu}_\theta v_\theta^h + 2\epsilon \right| &= \left| \sum_{\theta \in \widetilde{\Theta}} \sum_{h=1}^{d} \frac{N_\theta \gamma_\theta^h}{p \nu_h} + 2\epsilon \right| \\
&= \left| \frac{\sum_{\theta \in \widetilde{\Theta}} \sum_{h=1}^{d} \beta N_\theta \gamma_\theta^h \prod_{h' \neq h} \nu_{h'} + 2\alpha p \prod_{h=1}^{d} \nu_h}{p \prod_{h=1}^{d} \nu_h \beta} \right| \\
&\geq 2^{-B_{\widehat{\mu}} - B_\epsilon - 9d^3(B + B_\epsilon + B_{\widehat{\mu}})}.
\end{aligned}$$

Thus, the distance between $x^\diamond$ and any hyperplane H defining the boundary of $\mathcal{P}$ can be lower bounded as follows:

$$d(x^\diamond, H) \geq \frac{1}{d^2} 2^{-9d^3(B + B_\epsilon + B_{\widehat{\mu}}) - 4dB - B_{\widehat{\mu}} - B_\epsilon} \geq \frac{1}{d^2} 2^{-10d^3(B + B_\epsilon + B_{\widehat{\mu}})}.$$

We observe that, given the definition of ρ at Line 3, the distance $d(x^\diamond, x)$ is strictly smaller than $d(x^\diamond, H)$, showing that $x \in \text{int}(\mathcal{P})$. Furthermore, we can prove that the bit-complexity of x is bounded by

$$\mathcal{O}(d^3(B + B_\epsilon + B_{\widehat{\mu}}) + \log(1/\delta)).$$

To do so, we observe that the denominator of $x^\diamond$ is equal to $d \prod_{h=1}^{d} \nu_h$, while the denominator of y_i is equal to $M = \lceil \sqrt{d}/\delta \rceil$. As a result, the denominator of every $x_i = x_i^\circ + \rho y_i$, with $i \in [d-1]$, can be written as follows:

$$D = d \prod_{h=1}^{d} \nu_h D_\rho M,$$

where D_ρ is the denominator of the rational number ρ. Similarly, the last component x_d can be written with the same denominator. As a result, the bit complexity of $x \in [0,1]^d$ can be upper bounded as follows:

$$\begin{aligned}
B_x &\leq 2 \lceil \log(D) \rceil \\
&= \mathcal{O}(\log(d \prod_{h=1}^{d} \nu_h D_\rho M)) \\
&= \mathcal{O}\left(\log\left(\prod_{h=1}^{d} 2^{9d^2(B + B_\epsilon + B_{\widehat{\mu}})} \right) + \log(d 2^{10d^3(B + B_\epsilon + B_{\widehat{\mu}})}) + \log(\sqrt{d}/\delta) \right) \\
&= \mathcal{O}\left(d^3(B + B_\epsilon + B_{\widehat{\mu}}) + \log\left(\frac{1}{\delta}\right) \right).
\end{aligned}$$

Finally, with the same analysis performed in Lemma 3.8, we can show that the probability that x belongs to a given hyperplane H is at most δ. □

Lemma B.3. *Each vertex v of a polytope $\mathcal{P} \subseteq \mathcal{X}_\epsilon : \text{vol}_{d-1}(\mathcal{P}) > 0$, defined by separating or boundary hyperplanes, has bit-complexity at most $9d^2(B + B_\epsilon + B_{\widehat{\mu}})$. Furthermore, with a bit-complexity of $9d^2(B + B_\epsilon + B_{\widehat{\mu}})$, all the components of the vector v identifying a vertex can be written as fractions with the same denominator.*

Proof. We follow a line of reasoning similar to the proof of Lemma A.2. Let v be a vertex of the polytope $\mathcal{P}$. Then such a vertex lies on the hyperplane H' ensuring that the sum of its components is equal to one. Furthermore, it also belongs to a subset of $d-1$ linearly independent hyperplanes. These can be separating hyperplanes:

$$H_{ij} = \left\{ x \in \mathbb{R}^d \mid \sum_{\theta \in \Theta} \mu_\theta x_\theta (u_\theta(a_i) - u_\theta(a_j)) = 0 \right\}$$

with $a_i, a_j \in \mathcal{A}$, boundary hyperplanes of the form $H_i = \{x \in \mathbb{R}^d \mid x_i > 0\}$, or the hyperplane $\widehat{H} \coloneqq \left\{x \in \mathbb{R}^d \mid \sum_{\theta \in \widetilde{\Theta}} \widehat{\mu}_\theta x_\theta \geq 2\epsilon\right\}$. Consequently, there exists a matrix $A \in \mathbb{Q}^{d \times d}$ and a vector $b \in \mathbb{Q}^d$ such that $Av = b$. Suppose that v is not defined by the hyperplane

$$\widehat{H} \coloneqq \left\{ x \in \mathbb{R}^d \mid \sum_{\theta \in \Theta} \widehat{\mu}_\theta x_\theta \geq 2\epsilon \right\}.$$

Then, each entry of the matrix A is either equal to one or the quantity $\mu_\theta(u_\theta(a) - u_\theta(a'))$ for some $\theta \in \Theta$ and $a, a' \in \mathcal{A}$. Thus, its bit-complexity is bounded by B. Similarly, each entry of the vector b is either equal to one or zero. With a reasoning similar to the one applied in Chapter 3, the bit-complexity of v is at most $9d^2(B_\mu + B_u)$.

Suppose instead that v is defined also by the hyperplane $\widehat{H}$, corresponding to the last row of the matrix A and the last component of the vector b. This hyperplane can be rewritten as:

$$\widehat{H} = \left\{ x \in \mathbb{R}^d \mid \sum_{\theta \in \widetilde{\Theta}} \frac{\widehat{\mu}_\theta}{2\epsilon} x_\theta \geq 1 \right\}.$$

Thus, each element of the last row of A is either zero or the quantity $\widehat{\mu}_\theta / 2\epsilon$ for some $\theta \in \widetilde{\Theta}$. We observe that $\widehat{\mu}_\theta / 2\epsilon$ is a rational number with numerator bounded by $2^{B_{\widehat{\mu}} + B_\epsilon}$ and denominator bounded by $2^{B_{\widehat{\mu}} + B_\epsilon + 1}$. Thus, we multiply the last row of A and the last component of b by a constant bounded by $2^{d(B_{\widehat{\mu}} + B_\epsilon + 1)}$. The other rows of A and the corresponding components of b are multiplied instead by some constants bounded by 2^{4dB}. This way, we obtain an equivalent system $A'v = b'$ with integer coefficients.

We define $A'(j)$ as the matrix obtained by substituting the j-th column of A' with b'. Then, by Cramer's rule, the value of the j-th component of v_j can be

computed as follows:

$$v_j = \frac{\det(A'(j))}{\det(A')} \quad \forall j \in [d].$$

We observe that both determinants are integer numbers as the entries of both A' and b' are all integers, thus by Hadamard's inequality we have:

$$\begin{aligned}
|\det(A')| &\leq \prod_{i\in[d]} \sqrt{\sum_{j\in[d]} {a'_{ji}}^2} \\
&\leq \left(\prod_{i\in[d-1]} \sqrt{\sum_{j\in[d]} (2^{4dB})^2} \right) \sqrt{\sum_{j\in[d]} (2^{d(B_{\widehat{\mu}}+B_\epsilon+1)})^2} \\
&= \left(\prod_{i\in[d-1]} \sqrt{d(2^{4dB})^2} \right) \sqrt{d 2^{2d(B_{\widehat{\mu}}+B_\epsilon+1)}} \\
&= \left(\prod_{i\in[d-1]} d^{\frac{1}{2}}(2^{4dB}) \right) d^{\frac{1}{2}} 2^{d(B_{\widehat{\mu}}+B_\epsilon+1)} \\
&= d^{\frac{d}{2}}(2^{4d(d-1)B}) 2^{d(B_{\widehat{\mu}}+B_\epsilon+1)} \\
&\leq d^{\frac{d}{2}}(2^{4d^2B}) 2^{d(B_{\widehat{\mu}}+B_\epsilon+1)}.
\end{aligned}$$

With a reasoning similar to Chapter 3, we can show that that the bit-complexity D_v of the vertex v is bounded by:

$$D_v \leq 9Bd^2 + 2(d(B_{\widehat{\mu}} + B_\epsilon + 1)) \leq 9d^2(B + B_\epsilon + B_{\widehat{\mu}}).$$

Furthermore, this result holds when the denominator of every component v_j of the vertex v is written with the same denominator $\det(A')$, concluding the proof. □

APPENDIX C

Additional technical lemmas from Chapter 5

In this section, we further describe the estimators and confidence bounds for the stochastic quantities involved in an episodic MPP, namely, transitions, priors, sender's rewards, and receivers' ones.

C.0.1 Transition probabilities

First, we introduce confidence bounds for transition probabilities $P(x'|x,\theta,a)$, by generalizing those introduced by Rosenberg and Mansour [2019] for MDPs to MPPs. In the following, we let $N_t(x,\theta,a)$, respectively $N_t(x,\theta,a,x')$, be the counter specifying the number of episodes up to episode $t \in [T]$ (excluded) in which the triplet (x,θ,a), respectively the tuple (x,θ,a,x'), is visited. Then, the estimated probability of going from $x \in X$ to $x' \in X$ by playing action $a \in A$, when the outcome realized in state x is $\theta \in \Theta$, is defined as follows:

$$\overline{P}_t\left(x'|x,\theta,a\right) := \frac{N_t(x,\theta,a,x')}{\max\left\{1, N_t(x,\theta,a)\right\}}.$$

For any $\delta \in (0,1)$, the confidence set at episode $t \in [T]$ for the transition function P is $\mathcal{P}_t := \bigcap_{(x,\theta,a)\in X\times\Theta\times A} \mathcal{P}_t^{x,\theta,a}$, where $\mathcal{P}_t^{x,\theta,a}$ is a set of transition functions defined as:

$$\mathcal{P}_t^{x,\theta,a} := \left\{ \widehat{P} : \left\| \widehat{P}(\cdot|x,\theta,a) - \overline{P}_t(\cdot|x,\theta,a) \right\|_1 \leq \epsilon_t(x,\theta,a) \right\},$$

where $\widehat{P}(\cdot|x,\theta,a)$ and $\overline{P}_t(\cdot|x,\theta,a)$ are vectors whose entries are the values $\widehat{P}(x'|x,\theta,a)$ and $\overline{P}_t(x'|x,\theta,a)$, respectively, while $\epsilon_t(x,\theta,a)$ is a confidence bound defined as:

$$\epsilon_t(x,\theta,a) := \sqrt{\frac{2|X_{k(x)+1}| \ln\left(|T||X||\Theta||A|/\delta\right)}{\max\{1, N_t(x,\theta,a)\}}}.$$

The following lemma formally proves that $\mathcal{P}_t$ is a suitable confidence set for the transition function of an MPP.

Lemma C.1. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following condition holds for every $x \in X$, $\theta \in \Theta$, $a \in A$, and $t \in [T]$ jointly:*

$$\left\|P(\cdot|x,\theta,a) - \overline{P}_t(\cdot|x,\theta,a)\right\|_1 \leq \epsilon_t(x,\theta,a).$$

Lemma C.1 can be easily proven by employing a union bound over all x, θ, a, and t..

C.0.2 Prior distributions

Next, we introduce confidence bounds for prior distributions. For every state $x \in X$, we define $\overline{\mu}_t(\cdot|x) \in \Delta(\Theta)$ as the estimator of the prior distribution at x built by using observations up to episode $t \in [T]$ (this excluded). Formally, the entries of vector $\overline{\mu}_t(\cdot|x)$ are such that, for every $\theta \in \Theta$:

$$\overline{\mu}_t(\theta|x) := \frac{\sum_{\tau \in [t-1]} \mathbb{1}_\tau\{x,\theta\}}{\max\{1, N_t(x)\}},$$

where $N_t(x)$ is the number of visits to state x up to episode t (excluded), while $\mathbb{1}_\tau\{x,\theta\}$ is an indicator function equal to 1 if and only if the pair (x,θ) is visited at episode τ. The following lemma provides confidence bounds for priors.

Lemma C.2. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following holds for all $x \in X$ and $t \in [T]$ jointly:*

$$\|\mu(\cdot|x) - \overline{\mu}_t(\cdot|x)\|_1 \leq \zeta_t(x),$$

where we let $\zeta_t(x) := \sqrt{\frac{2|\Theta| \ln(|T||X|/\delta)}{\max\{1, N_t(x)\}}}$.

Lemma C.2 follows by applying Bernstein's inequality and a union bound over all states and episodes.

C.0.3 Sender's and receivers' rewards

Finally, we introduce estimators for rewards. In the following, present the results related to sender's rewards and receiver's rewards under full and partial feedback. For every $x \in X$, $\theta \in \Theta$, and $a \in A$, the estimated sender's and receivers' rewards built with observations up to episode $t \in [T]$ (this excluded) are defined as follows:

$$\overline{r}_{S,t}(x,\theta,a) := \frac{\sum_{\tau \in [t-1]} r_{S,\tau}(x,\theta,a) \mathbb{1}_\tau\{x,\theta,a\}}{\max\{1, N_t(x,\theta,a)\}},$$

$$\overline{r}_{R,t}(x,\theta,a) := \frac{\sum_{\tau\in[t-1]} r_{R,\tau}(x,\theta,a)\mathbb{1}_\tau\{x,\theta,a\}}{\max\{1, N_t(x,\theta,a)\}},$$

where $\mathbb{1}_\tau\{x,\theta,a\}$ is an indicator function equal to 1 if and only if the triplet (x,θ,a) is visited during episode τ.

The following lemma provides confidence bounds for sender's rewards, when *full* feedback is available.

Lemma C.3. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following condition holds for every $x \in X$, $\theta \in \Theta$, $a \in A$, and $t \in [T]$ jointly:*

$$\left|r_S(x,\theta,a) - \overline{r}_{S,t}(x,\theta,a)\right| \le \xi_{S,t}(x,\theta,a),$$

where $\xi_{S,t}(x,\theta,a) := \min\left\{1, \sqrt{\frac{\ln(3T|X||\Theta|/\delta)}{\max\{1,N_t(x,\theta)\}}}\right\}$.

Lemma C.3 follows by applying Hoeffding's inequality and a union bound over all x, θ and t.

The following lemma provides confidence bounds for receiver's rewards, when *full* feedback is available.

Lemma C.4. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following condition holds for every $x \in X$, $\theta \in \Theta$, $a \in A$, and $t \in [T]$ jointly:*

$$\left|r_R(x,\theta,a) - \overline{r}_{R,t}(x,\theta,a)\right| \le \xi_{R,t}(x,\theta,a),$$

where $\xi_{R,t}(x,\theta,a) := \min\left\{1, \sqrt{\frac{\ln(3T|X||\Theta|/\delta)}{\max\{1,N_t(x,\theta)\}}}\right\}$.

Lemma C.4 follows by applying Hoeffding's inequality and a union bound over all x, θ and t.

The following lemma provides confidence bounds for sender's rewards, when only *partial* feedback is available.

Lemma C.5. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following condition holds for every $x \in X$, $\theta \in \Theta$, $a \in A$, and $t \in [T]$ jointly:*

$$\left|r_S(x,\theta,a) - \overline{r}_{S,t}(x,\theta,a)\right| \le \xi_{S,t}(x,\theta,a),$$

where $\xi_{S,t}(x,\theta,a) := \min\left\{1, \sqrt{\frac{\ln(3T|X||\Theta||A|/\delta)}{\max\{1,N_t(x,\theta,a)\}}}\right\}$.

Lemma C.5 follows by applying Hoeffding's inequality and a union bound over all x, θ, a, and t.

Finally, the following lemma provides confidence bounds for receiver's rewards, when only *partial* feedback is available.

Lemma C.6. *Given any $\delta \in (0,1)$, with probability at least $1-\delta$, the following condition holds for every $x \in X$, $\theta \in \Theta$, $a \in A$, and $t \in [T]$ jointly:*

$$\left|r_R(x,\theta,a) - \overline{r}_{R,t}(x,\theta,a)\right| \le \xi_{R,t}(x,\theta,a),$$

where $\xi_{R,t}(x,\theta,a) := \min\left\{1, \sqrt{\frac{\ln(3T|X||\Theta||A|/\delta)}{\max\{1,N_t(x,\theta,a)\}}}\right\}$.

Lemma C.6 follows by applying Hoeffding's inequality and a union bound over all x, θ, a, and t.

Bibliography

Shipra Agrawal, Yiding Feng, and Wei Tang. Dynamic pricing and learning with bayesian persuasion. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 59273–59285. Curran Associates, Inc., 2023.

Rui Ai, Chang Wang, Chenchen Li, Jinshan Zhang, Wenhan Huang, and Xiaotie Deng. No-regret learning in repeated first-price auctions with budget constraints, 2022. URL https://arxiv.org/abs/2205.14572.

Tal Alon, Paul Dütting, and Inbal Talgam-Cohen. Contracts with private cost per unit-of-effort. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 52–69, 2021.

Ricardo Alonso and Odilon Câmara. Persuading voters. *American Economic Review*, 2016.

E. Amaldi, A. Capone, S. Coniglio, and L. G. Gianoli. Network optimization problems subject to max–min fair flow allocation. *IEEE Communications Letters*, 17(7): 1463–1466, 2013. doi: 10.1109/LCOMM.2013.051313.122950.

Itai Arieli and Yakov Babichenko. Private bayesian persuasion. *Journal of Economic Theory*, 182:185–217, 2019.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2-3):235–256, 2002. URL http://homes.dsi.unimi.it/~cesabian/Pubblicazioni/ml-02.pdf.

Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, Giulia Romano, Nicola Gatti, et al. Public signaling in bayesian ad auctions. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 39–45, 2022.

Francesco Bacchiocchi, Matteo Bollini, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Online bayesian persuasion without a clue. In *Advances in Neural Information Processing Systems*, volume 37, pages 76404–76449, 2024a.

Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Learning optimal contracts: How to exploit small action spaces. In *12th International Conference on Learning Representations, ICLR 2024*, 2024b.

Francesco Bacchiocchi, Matteo Bollini, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. The sample complexity of stackelberg games. In *Proceedings of The 28th International Conference on Artificial Intelligence and Statistics*, pages 2053–2061, 2025a.

Francesco Bacchiocchi, Matteo Castiglioni, Nicola Gatti, and Alberto Marchesi. Learning optimal contracts with small action spaces. *Artificial Intelligence*, page 104334, 2025b.

Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Regret minimization for piecewise linear rewards: Contracts, auctions, and beyond. In *Proceedings of the 26th ACM Conference on Economics and Computation*, pages 1020–1020, 2025c.

Francesco Bacchiocchi, Jiarui Gan, Matteo Castiglioni, Alberto Marchesi, Nicola Gatti, et al. Contract design under approximate best responses. In *Forty-Second International Conference on Machine Learning*, 2025d.

Francesco Bacchiocchi, Francesco Emanuele Stradi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Markov persuasion processes: Learning to persuade from scratch. In *Advances in Neural Information Processing Systems*, 2025e.

Ashwinkumar Badanidiyuru, Kshipra Bhawalkar, and Haifeng Xu. Targeting and signaling in ad auctions. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2018.

Yu Bai, Chi Jin, Huan Wang, and Caiming Xiong. Sample-efficient learning of stackelberg equilibria in general-sum games. *Advances in Neural Information Processing Systems*, 34:25799–25811, 2021.

Maria-Florina Balcan, Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, EC '15, page 61–78, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450334105. doi: 10.1145/2764468.2764478. URL https://doi.org/10.1145/2764468.2764478.

Keith Ball. Cube slicing in rn. *Proceedings of the American Mathematical Society*, 97 (3):465–473, 1986. ISSN 00029939, 10886826.

Hamsa Bastani, Mohsen Bayati, Mark Braverman, Ramki Gummadi, and Ramesh Johari. Analysis of medicare pay-for-performance contracts. *Available at SSRN 2839143*, 2016.

Martino Bernasconi, Matteo Castiglioni, Alberto Marchesi, Nicola Gatti, and Francesco Trovò. Sequential information design: Learning to persuade in the dark. *Advances in Neural Information Processing Systems*, 35:15917–15928, 2022.

Martino Bernasconi, Matteo Castiglioni, Andrea Celli, Alberto Marchesi, Francesco Trovò, and Nicola Gatti. Optimal rates and efficient algorithms for online Bayesian persuasion. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 2164–2183. PMLR, 23–29 Jul 2023a.

Martino Bernasconi, Matteo Castiglioni, Alberto Marchesi, and Mirco Mutti. Persuading farsighted receivers in MDPs: the power of honesty. In *Advances in Neural Information Processing Systems*, volume 36, pages 1–13, 2023b.

Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

Umang Bhaskar, Yu Cheng, Young Kun Ko, and Chaitanya Swamy. Hardness results for signaling in bayesian zero-sum and network routing games. In *Proceedings of the 2016 ACM Conference on Economics and Computation* (*EC '16*), pages 479–496, Maastricht, The Netherlands, 2016. ACM.

Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Learning optimal commitment to overcome insecurity. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'14, page 1826–1834, Cambridge, MA, USA, 2014. MIT Press.

Patrick Bolton and Mathias Dewatripont. Contract theory. *MIT Press Books*, 1, 2005.

Sébastien Bubeck, Gilles Stoltz, Csaba Szepesvári, and Rémi Munos. Online optimization in x-armed bandits. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008.

Federico Cacciamani, Matteo Castiglioni, and Nicola Gatti. Online mechanism design for information acquisition. In *International Conference on Machine Learning*, pages 3299–3326. PMLR, 2023.

Modibo K. Camara, Jason D. Hartline, and Aleck Johnsen. Mechanisms for a no-regret agent: Beyond the common prior. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science* (*FOCS*), pages 259–270, 2020.

Gabriel Carroll. Robustness and linear contracts. *American Economic Review*, 105 (2):536–563, 2015.

Matteo Castiglioni, Andrea Celli, and Nicola Gatti. Persuading voters: It's easy to whisper, it's hard to speak loud. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence* (*AAAI-20*), pages 1870–1877, New York, NY, USA, 2020a. AAAI Press. doi: 10.1609/aaai.v34i02.5555.

Matteo Castiglioni, Andrea Celli, Alberto Marchesi, and Nicola Gatti. Online bayesian persuasion. *Advances in Neural Information Processing Systems*, 33:16188–16198, 2020b.

Matteo Castiglioni, Andrea Celli, Alberto Marchesi, and Nicola Gatti. Signaling in bayesian network congestion games: the subtle power of symmetry. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5252–5259, 2021a.

Matteo Castiglioni, Alberto Marchesi, Andrea Celli, and Nicola Gatti. Multi-receiver online bayesian persuasion. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 1314–1323. PMLR, 18–24 Jul 2021b.

Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Bayesian agency: Linear versus tractable contracts. *Artificial Intelligence*, 307:103684, 2022.

Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Designing menus of contracts efficiently: The power of randomization. *Artificial Intelligence*, 318:103881, 2023.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Nicolo Cesa-Bianchi, Tommaso Cesari, and Vianney Perchet. Dynamic pricing with finitely many unknown valuations. In *Algorithmic Learning Theory*, pages 247–273. PMLR, 2019.

Yurong Chen, Zhaohua Chen, Xiaotie Deng, and Zhiyi Huang. Are bounded contracts learnable and approximately optimal? *arXiv preprint arXiv:2402.14486*, 2024.

Aviad Cohen, Aris Deligkas, and Michael Koren. Learning approximately optimal contracts. In *Algorithmic Game Theory: 15th International Symposium, SAGT 2022, Colchester, UK, September 12–15, 2022, Proceedings*, volume 13492 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2022.

Lee Cohen and Yishay Mansour. Optimal algorithm for bayesian incentive-compatible exploration. In *Proceedings of the 2019 ACM Conference on Economics and Computation (EC '19)*, pages 135–151, Phoenix, AZ, USA, 2019. ACM.

Lin William Cong and Zhiguo He. Blockchain disruption and smart contracts. *The Review of Financial Studies*, 32(5):1754–1797, 2019.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, 2006.

Paul Duetting, Guru Guruganesh, Jon Schneider, and Joshua Ruizhi Wang. Optimal no-regret learning for one-sided Lipschitz functions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8836–8850. PMLR, 23–29 Jul 2023a.

Paul Duetting, Guru Guruganesh, Jon Schneider, and Joshua Ruizhi Wang. Optimal no-regret learning for one-sided Lipschitz functions. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8836–8850. PMLR, 23–29 Jul 2023b.

Shaddin Dughmi and Haifeng Xu. Algorithmic bayesian persuasion. In *Proceedings of the 48th Annual ACM Symposium on Theory of Computing (STOC)*, pages 412–425. ACM, 2016.

Shaddin Dughmi and Haifeng Xu. Algorithmic persuasion with no externalities. In *Proceedings of the 2017 ACM Conference on Economics and Computation (EC)*, pages 351–368. ACM, 2017.

Paul Dütting, Tim Roughgarden, and Inbal Talgam-Cohen. Simple versus optimal contracts. In *Proceedings of the 2019 ACM Conference on Economics and Computation*, pages 369–387, 2019.

Paul Dutting, Tim Roughgarden, and Inbal Talgam-Cohen. The complexity of contracts. *SIAM Journal on Computing*, 50(1):211–254, 2021.

Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Combinatorial contracts. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 815–826. IEEE, 2022.

Paul Dütting, Tomer Ezra, Michal Feldman, and Thomas Kesselheim. Multi-agent contracts. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 1311–1324, 2023.

Paul Dutting, Michal Feldman, and Yoav Gal Tzur. Combinatorial contracts beyond gross substitutes. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 92–108. SIAM, 2024.

Paul Dütting, Michal Feldman, Inbal Talgam-Cohen, et al. Algorithmic contract theory: A survey. *Foundations and Trends® in Theoretical Computer Science*, 16(3-4):211–412, 2024.

Yuval Emek, Michal Feldman, Iftah Gamzu, Renato PaesLeme, and Moshe Tennenholtz. Signaling schemes for revenue maximization. *ACM Transactions on Economics and Computation (TEAC)*, 2(2):1–19, 2014.

Yiding Feng, Wei Tang, and Haifeng Xu. Online bayesian recommendation with no regret. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, page 818–819, 2022.

Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In Hal Daumé III and Aarti Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 3133–3144. PMLR, 13–18 Jul 2020.

Michal Forišek. Approximating rational numbers by fractions. In Pierluigi Crescenzi, Giuseppe Prencipe, and Geppino Pucci, editors, *Fun with Algorithms*, volume 4475 of *Lecture Notes in Computer Science*, pages 156–165, Berlin, Heidelberg, 2007. Springer. ISBN 978-3-540-72914-3.

Jiarui Gan, Minbiao Han, Jibang Wu, and Haifeng Xu. Generalized principal-agency: Contracts, information, games and beyond. *arXiv preprint arXiv:2209.01146*, 2022a. URL https://arxiv.org/abs/2209.01146. Revised February 2024.

Jiarui Gan, Rupak Majumdar, Goran Radanovic, and Adish Singla. Bayesian persuasion in sequential decision-making. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 5025–5033, 2022b.

Jiarui Gan, Minbiao Han, Jibang Wu, and Haifeng Xu. Robust stackelberg equilibria. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 735, New York, NY, USA, 2023a. Association for Computing Machinery. ISBN 9798400701047.

Jiarui Gan, Rupak Majumdar, Debmalya Mandal, and Goran Radanovic. Sequential principal-agent problems with communication: Efficient computation and learning, 2023b.

Itay Goldstein and Yaron Leitner. Stress tests and information disclosure. *Journal of Economic Theory*, 177:34–69, 2018.

Sanford J Grossman and Oliver D Hart. An analysis of the principal-agent problem. In *Foundations of Insurance Economics: Readings in Economics and Finance*, pages 302–340. Springer, 1992.

Guru Guruganesh, Jon Schneider, and Joshua R Wang. Contracts under moral hazard and adverse selection. In *Proceedings of the 22nd ACM Conference on Economics and Computation*, pages 563–582, 2021.

Guru Guruganesh, Jon Schneider, Joshua Wang, and Junyao Zhao. The power of menus in contract design. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 818–848, 2023.

Nika Haghtalab, Fei Fang, Thanh H. Nguyen, Arunesh Sinha, Ariel D. Procaccia, and Milind Tambe. Three strategies to success: learning adversary models in security games. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, page 308–314. AAAI Press, 2016. ISBN 9781577357704.

Minbiao Han, Michael Albert, and Haifeng Xu. Learning in online principal-agent interactions: the power of menus. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*. AAAI Press, 2024a. ISBN 978-1-57735-887-9.

Yanjun Han, Tsachy Weissman, and Zhengyuan Zhou. Optimal no-regret learning in repeated first-price auctions. *Operations Research*, 2024b.

Chien-Ju Ho, Aleksandrs Slivkins, and Jennifer Wortman Vaughan. Adaptive contract design for crowdsourcing markets: Bandit algorithms for repeated principal-agent problems. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, page 359–376, 2014.

Bengt Holmstrom and Paul Milgrom. Multitask principal–agent analyses: Incentive contracts, asset ownership, and job design. *The Journal of Law, Economics, and Organization*, 7(special_issue):24–52, 1991.

Grigory Ivanov. On the volume of projections of the cross-polytope. *Discrete Mathematics*, 344(5):112312, 2021. ISSN 0012-365X.

Emir Kamenica and Matthew Gentzkow. Bayesian persuasion. *American Economic Review*, 101(6):2590–2615, 2011.

Robert Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, 17, 2004.

Robert Kleinberg and Tom Leighton. The value of knowing a demand curve: Bounds on regret for online posted-price auctions. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 594–605. IEEE, 2003.

Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Multi-armed bandits in metric spaces. In *Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing*, STOC '08, page 681–690. Association for Computing Machinery, 2008. ISBN 9781605580470.

Martine Labbé and Anthony Violin. Bilevel programming and price setting problems. *Annals of Operations Research*, 240(1):141–169, 2016. doi: 10.1007/s10479-014-1584-2.

Jean-Jacques Laffont and David Martimort. The theory of incentives: the principal-agent model. In *The theory of incentives*. Princeton university press, 2009.

Niklas T. Lauffer, Mahsa Ghasemi, Abolfazl Hashemi, Yagiz Savas, and Ufuk Topcu. No-regret learning in dynamic stackelberg games. *ArXiv*, abs/2202.04786, 2022.

Joshua Letchford, Vincent Conitzer, and Kamesh Munagala. Learning and approximating the optimal strategy to commit to. In *Algorithmic Game Theory: Second International Symposium, SAGT 2009, Paphos, Cyprus, October 18-20, 2009. Proceedings 2*, pages 250–262. Springer, 2009.

Tao Lin and Yiling Chen. Generalized principal-agent problem with a learning agent. In *International Conference on Learning Representations*, 2025.

Tao Lin and Ce Li. Information design with unknown prior. In *16th Innovations in Theoretical Computer Science Conference (ITCS 2025)*, pages 72–1. Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2025.

Yishay Mansour, Alex Slivkins, Vasilis Syrgkanis, and Zhiwei Steven Wu. Bayesian exploration: Incentivizing exploration in Bayesian games. *Operations Research*, 70 (2):1105–1127, 2022.

Jieming Mao, Renato Leme, and Jon Schneider. Contextual pricing for lipschitz buyers. In *Advances in Neural Information Processing Systems*, 2018.

Andreu Mas-Colell, Michael D. Whinston, and Jerry R. Green. *Microeconomic Theory*. Oxford University Press, New York, 1995.

Nishant A. Mehta, Junpei Komiyama, Vamsi K. Potluru, Andrea Nguyen, and Mica Grant-Hagen. Thresholded linear bandits. In *Proceedings of The 26th International Conference on Artificial Intelligence and Statistics*, volume 206 of *Proceedings of Machine Learning Research*, pages 6968–7020. PMLR, 25–27 Apr 2023.

Peter Bro Miltersen and Or Sheffet. Send mixed signals: Earn more, work less. In *Proceedings of the Thirteenth ACM Conference on Electronic Commerce (EC '12)*, pages 234–247, Valencia, Spain, June 2012. ACM. doi: 10.1145/2229012.2229033.

Roger B Myerson. Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81, 1982.

Francesco Orabona. A modern introduction to online learning. *CoRR*, abs/1912.13213, 2019.

Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: An efficient exact algorithm for solving bayesian stackelberg games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 895–902. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

Binghui Peng, Weiran Shen, Pingzhong Tang, and Song Zuo. Learning optimal strategies to commit to. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 2149–2156, 2019.

Zinovi Rabinovich, Albert Xin Jiang, Manish Jain, and Haifeng Xu. Information disclosure as a means to security. In *Proceedings of the Fourteenth International Conference on Autonomous Agents and Multiagent Systems (AAMAS '15)*, pages 645–653, Istanbul, Turkey, 2015. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Aviv Rosenberg and Yishay Mansour. Online convex optimization in adversarial Markov decision processes. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5478–5486. PMLR, 09–15 Jun 2019.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. ISBN 0471908541.

Steven Shavell. Risk sharing and incentives in the principal and agent relationship. *The Bell Journal of Economics*, pages 55–73, 1979.

Milind Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, Cambridge, 2011. ISBN 978-1-107-00677-4. doi: 10.1017/CBO9780511973031.

Shoshana Vasserman, Michal Feldman, and Avinatan Hassidim. Implementing the wisdom of waze. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI 2015)*, 2015.

Heinrich von Stackelberg. *Marktform und Gleichgewicht*. Julius Springer, Wien und Berlin, 1934. ISBN 978-3-642-12585-0. Translated as *Market Structure and Equilibrium*, Springer, 2011.

Bernhard von Stengel and Shmuel Zamir. Leadership games with convex strategy sets. *Games and Economic Behavior*, 69(2):446–457, 2010.

Jibang Wu, Zixuan Zhang, Zhe Feng, Zhaoran Wang, Zhuoran Yang, Michael I. Jordan, and Haifeng Xu. Sequential information design: Markov persuasion process and its efficient reinforcement learning. In *Proceedings of the 23rd ACM Conference on Economics and Computation (EC '22)*. ACM, 2022.

Haifeng Xu. On the tractability of public persuasion with no externalities. In *Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2708–2727. SIAM, 2020.

Haifeng Xu, Rupert Freeman, Vincent Conitzer, Shaddin Dughmi, and Milind Tambe. Signaling in bayesian stackelberg games. In *Proceedings of the 2016 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '16)*, pages 150–158, Singapore, 2016. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS).

Kunhe Yang and Hanrui Zhang. Computational aspects of bayesian persuasion under approximate best response. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, 2025.

Banghua Zhu, Stephen Bates, Zhuoran Yang, Yixin Wang, Jiantao Jiao, and Michael I Jordan. The sample complexity of online contract design. In *Proceedings of the 24th ACM Conference on Economics and Computation*, pages 1188–1188, 2023.

You Zu, Krishnamurthy Iyer, and Haifeng Xu. Learning to persuade on the fly: Robustness against ignorance. In *Proceedings of the 22nd ACM Conference on Economics and Computation (EC)*, pages 927–928, 2021.