The Sample Complexity of Stackelberg Games

Francesco Bacchiocchi, Matteo Bollini, Matteo Castiglioni, Alberto Marchesi, Nicola Gatti
Politecnico di Milano

Abstract

Stackelberg games (SGs) constitute the most fundamental and acclaimed models of strategic interactions involving some form of commitment. Moreover, they form the basis of more elaborate models of this kind, such as, *e.g.*, Bayesian persuasion and principal-agent problems. Addressing *learning* tasks in SGs and related models is crucial to operationalize them in practice, where model parameters are usually unknown. In this paper, we revise the *sample complexity* of learning an optimal strategy to commit to in SGs. We provide a novel algorithm that (i) does *not* require any of the limiting assumptions made by state-of-the-art approaches and (ii) deals with a trade-off between sample complexity and termination probability arising when leader's strategies representation has finite precision. Such a trade-off has been completely neglected by existing algorithms and, if *not* properly managed, it may result in them using exponentially-many samples. Our algorithm requires novel techniques, which also pave the way to addressing learning problems in other models with commitment ubiquitous in the real world.

1 INTRODUCTION

Asymmetries are ubiquitous in strategic interactions that involve multiple agents. The most fundamental and acclaimed models of asymmetric interactions are *Stackelberg games* (SGs) (Von Stackelberg, 1934; Conitzer and Sandholm, 2006; Von Stengel and Zamir, 2010). In an SG, a *leader* has the ability to publicly *commit to a strategy* beforehand, while a *fol-*

lower reacts by best responding to it. Such a simple idea of commitment is at the core of several other (more elaborate) models of asymmetric strategic interactions, such as Bayesian persuasion (Kamenica and Gentzkow, 2011), principal-agent problems (Myerson, 1982), and mechanism design (Myerson, 1989).

Recently, models of strategic interactions that involve some form of commitment have received a growing attention. Addressing *learning* tasks related to such models is of paramount importance to operationalize them in practice, where model parameters are usually *unknown*. Several works pursued this goal in various settings, ranging from SGs (Letchford et al., 2009; Blum et al., 2014; Balcan et al., 2015; Blum et al., 2019; Peng et al., 2019; Fiez et al., 2020; Bai et al., 2021; Lauffer et al., 2022) to Bayesian persuasion (Castiglioni et al., 2020; Zu et al., 2021) and principal-agent problems (Ho et al., 2014; Cohen et al., 2022; Zhu et al., 2023; Bacchiocchi et al., 2023).

In this paper, we revise the *sample complexity* of learning an optimal strategy to commit to in SGs. Letchford et al. (2009) first addressed this learning problem, by providing an algorithm that works by "sampling" suitably-selected leader's strategies to get information about follower's best responses. The main drawback of such an algorithm 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 top of the results by Letchford et al. (2009) to design an algorithm requiring a number of samples growing polynomially in L and exponentially in either m or the number of follower's actions n. Moreover, Peng et al. (2019) provide a lower bound showing that such a sample complexity result is tight.

The algorithm by Peng et al. (2019) relies on a number of assumptions that limit its applicability in practice. Specifically, it assumes to have control over the action played by the follower when multiple best responses are available to them, and, additionally, that follower's payoffs satisfy some suitable non-degeneracy conditions. However, we show that there exists a few

other corner cases that may lead to a failure of this algorithm. Furthermore, we aim at removing these assumptions in order to address every possible SG.

Another issue 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. Unless an opportune oracle is available, this may result in the algorithm requiring an exponential number of samples, as we show in this paper. Properly accounting for the representation precision of leader's strategies requires managing a challenging trade-off between the number of samples used by the algorithm and the probability with which it terminates. Understanding how to deal with such a trade-off is fundamental, *not* only for the problem of learning optimal commitments in SGs, but also to build a solid basis to tackle related learning problems in other models involving commitment.

In this paper, we introduce a new algorithm to learn an optimal strategy to commit to in SGs. Our algorithm requires a number of samples that scales polynomially in L and exponentially in either m or n—this is tight due to the lower bound by Peng et al. (2019)—, it does *not* require any of the assumptions made by Peng et al. (2019), and it circumvents all the issues of their algorithmic approach by properly managing the representation precision of leader's strategies.

2 PRELIMINARIES

A *(normal-form) Stackelberg game* (SG) is a tuple $G := (A_\ell, A_f, u_\ell, u_f)$ where: $A_\ell := \{a_i\}_{i=1}^m$ is a finite set of m leader's actions, $A_f := \{a_j\}_{j=1}^n$ is a finite set of n follower's actions, while $u_\ell, u_f : A_\ell \times A_f \to \mathbb{Q} \cap [0,1]$ are leader's and follower's utility functions, respectively. Specifically, $u_\ell(a_i, a_j)$ and $u_f(a_i, a_j)$ encode the payoffs obtained by the players when the leader plays $a_i \in A_\ell$ and the follower plays $a_j \in A_f$. A leader's *mixed strategy* is a probability distribution $p \in \Delta_m$ over leader's actions, with p_i denoting the probability of action $a_i \in A_\ell$. The space of all leader's mixed strategies is the $(m-1)$-dimensional simplex, namely $\Delta_m := \{p \in \mathbb{R}_+^m \mid \sum_{a_i \in A_\ell} p_i = 1\}$.

In an SG, the leader commits to a mixed strategy beforehand, and the follower decides how to play after observing it. Given a leader's commitment $p \in \Delta_m$, we assume w.l.o.g. that the follower plays an action deterministically. In particular, the follower plays a *best response*, *i.e.*, an action maximizing their expected utility under p. The set of follower's best responses is

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

As customary in the literature on SGs (Conitzer and Sandholm, 2006), we assume that the follower breaks ties in favor of the leader when having multiple best responses available, choosing an action maximizing leader's expected utility. Formally, after observing a leader's commitment $p \in \Delta_m$, the follower plays an action $a_f^\star(p) \in A_f$ such that

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

In an SG, the goal of the leader is to find an *optimal strategy to commit to*, which is one maximizing their expected utility given that the follower always reacts with a best response. Formally, the leader faces the bilevel optimization problem $\max_{p \in \Delta_m} u_\ell(p)$, in which, for ease of notation, we let

$$u_\ell(p) := \sum_{a_i \in A_\ell} p_i u_\ell(a_i, a_f^\star(p))$$

be leader's expected utility by committing to $p \in \Delta_m$.

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 $a_j, a_k \in A_f : a_j \neq a_k$, 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:

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

Moreover, we denote by $H_{jk} := \partial \mathcal{H}_{jk}$ the hyperplane constituting the boundary of the halfspace $\mathcal{H}_{jk}$, which we call the *separating hyperplane* between a_j and a_k.[1] 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:

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

The family of all the sets $\mathcal{P}_j$ is a cover of Δ_m, since it holds $\Delta_m = \bigcup_{a_j \in A_f} \mathcal{P}_j$. 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

$$H_i := \{p \in \mathbb{R}^m \mid p_i = 0\}$$

---

[1] 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 paper, we use both names for ease of presentation.

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 $\binom{n+m}{m}$. 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.[2]

Once all the best-response regions are available, the optimization problem faced by the leader can be formulated as $\max_{a_j \in A_f} \max_{p \in \mathcal{P}_j} u_\ell(p)$, where the inner max can be solved efficiently by means of linear programming (Conitzer and Sandholm, 2006).

Learning in Stackelberg games We study 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 $\mathtt{Oracle}(p)$, 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 $\mathtt{Oracle}(p)$. 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.

On the representation of numbers Throughout the paper, we assume that all the numbers manipulated by our algorithms are rational. We assume that rational numbers are represented as fractions, by specifying two integers encoding their numerator and denominator (Schrijver, 1986). Given a rational $q \in \mathbb{Q}$ 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 $B_{b/c} := B_b + B_c$, where B_b (B_c) is the number of bits required to represent the numerator (denominator). For ease of presentation and with an abuse of terminology, given a vector in $\mathbb{Q}^D$ of D ratio-

nals represented as fractions, we let its bit-complexity be the maximum bit-complexity among its entries.

3 STATE-OF-THE-ART OF LEARNING IN STACKELBERG GAMES

We start by presenting the algorithm by Peng et al. (2019), which is the state-of-the-art approach to learn an optimal strategy to commit to in SGs. We describe in detail the assumptions made by such an algorithm, and we showcase an SG in which the algorithm fails even if all its assumptions are met. Moreover, we discuss all the issues resulting from the fact that Peng et al. (2019) do *not* account for the bit-complexity of leader's strategies in their analysis.

3.1 The algorithm by Peng et al. (2019)

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 some *binary searches* 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 selects a strategy 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

---

[2] We denote by $\mathrm{vol}_d(\mathcal{P})$ the Lebesgue measure in d dimensions of a polytope $\mathcal{P} \subseteq \mathbb{R}^D$. For ease of notation, whenever $D = m$ and $d = m-1$, we simply write $\mathrm{vol}(\mathcal{P})$. Moreover, we let $\mathrm{int}(\mathcal{P})$ be the interior of $\mathcal{P}$ relative to a subspace that fully contains $\mathcal{P}$ and has minimum dimension. In the case of a best-response region $\mathcal{P}_j$, the $(m-1)$-dimensional simplex is one of such subspaces.

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.

3.2 Assumptions of Peng et al. (2019)

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 $\mathrm{vol}(\mathcal{P}_j)$ 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.

Assumption (a) imposes that no follower's action is a best response only for a "small" or zero-volume subset of leader's strategies. At the same time, assumptions (c) and (d) limit the set of SGs in which the algorithm can be applied by ruling out degenerate instances.

Assumption (b) is rather stronger than the typical tie-breaking rule adopted in the literature (see, *e.g.*, (Conitzer and Sandholm, 2006)). Furthermore, it is not easy to satisfy in practice, since it amounts to assuming that the algorithm has some form of control on which best responses are played by the follower. Similarly, it is not even possible to force the follower to play a particular action by slightly perturbating the leader's strategy. Due to the limited knowledge available, the algorithm does not necessarily know whether the perturbation is necessary, or how to construct it.

The algorithm by Peng et al. (2019) needs assumptions (a), (c), and (d) to meet the requirements of the procedure by Letchford et al. (2009), while assumption (b) is needed to properly build the lower bounds of best-response regions by vertex enumeration. As we show in the rest of this paper, our algorithm employs novel techniques that allow to drop assumptions (a)–(d).

Issues of the algorithm by Peng et al. (2019) Albeit assumptions (a)–(d) discard some challenging and degenerate instance, there exists a small set of SGs in which the Algorithm by Peng et al. (2019) fails to find an optimal commitment. Specifically, an issue may arise if a vertex of the simplex is also a vertex of two different best-response regions. In such a case, the procedure by Letchford et al. (2009) may be executed without meeting its requirements. An instance that may cause this failure is depicted in Figure 1. The algorithm fails since binary search finds a point p° on a boundary hyperplane. This clearly makes the
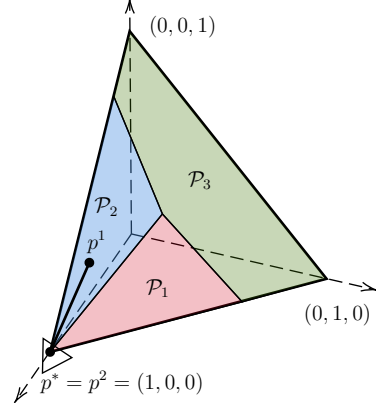

Figure 1: Algorithm by Peng et al. (2019) fails even if all its assumptions are met.

construction of a "small" $(m-1)$-dimensional simplex centered at p° impossible. Intuitively, this happens when the procedure by Letchford et al. (2009) randomly draws a leader's strategy from the intersection of the upper bounds of $\mathcal{P}_1$ and $\mathcal{P}_2$ obtaining a point p^1 in $\mathcal{P}_2$. Then, since at that point the lower bound of $\mathcal{P}_1$ only contains a point p^2 on a facet of $\mathcal{P}_1$, binary search is performed on a segment fully contained in $\mathcal{P}_2$. Thus, the search inevitably ends at $p^\circ = p^2$, resulting in a failure. Additional details are provided in Appendix B, where we also showcase another minor issue of the procedure by Letchford et al. (2009), inherited by Peng et al. (2019).

3.3 The algorithm by Peng et al. (2019) may require exponentially-many samples

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 paper, 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 2 shows an example of algorithm execution suffering from the issue described above. The complete example is provided in Appendix B. Intuitively, in Figure 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 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 paper, 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.
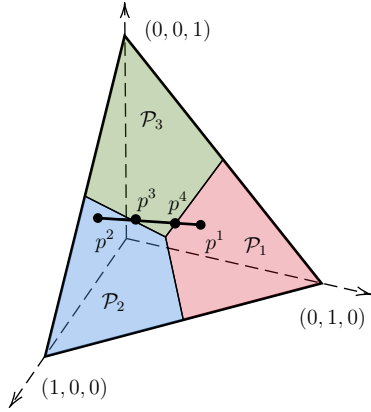

Figure 2: Algorithm by Peng et al. (2019) requires exponentially-many samples.

4   LEARNING AN OPTIMAL COMMITMENT

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 $\mathrm{vol}(\mathcal{P}_j) > 0$, 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 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 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 4.1 introduces the main procedure executed by the algorithm, called `Learn-Optimal-Commitment`. Sections 4.2 and 4.3 introduce `Find-Hyperplane` and `Binary-Search`, respectively, which are two procedures working in tandem to find separating hyperplanes. Finally, Section 4.4 describes `Sample-Int`, which is called any time the algorithm has to randomly sample a point in the interior of some polytope.

4.1   Learn-Optimal-Commitment

The pseudocode of `Learn-Optimal-Commitment` is provided in Algorithm 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.[3]

Algorithm 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

---

[3]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). As shown in Appendix F, our algorithm can be extended to also deal with such cases, and all our results continue to hold.

Algorithm 1 Learn-Optimal-Commitment

Require: Parameter $\zeta \in (0,1)$
1: $B \leftarrow$ Bit-complexity of Sample-Int $\quad \triangleright$ See Lemma 4.8
2: $\lambda \leftarrow m2^{-m(B+4L)-1}$ $\qquad\qquad\quad \triangleright$ See Lemma 4.1
3: $\delta \leftarrow \zeta/2(n^2+nm)^2+n^3$
4: $\mathcal{C} \leftarrow \varnothing$ $\qquad\qquad\quad \triangleright$ 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 $\text{int}(\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_k)$
7: $\quad a_j \leftarrow$ Oracle(p^{int})
8: $\quad \mathcal{U}_j \leftarrow \Delta_m$ $\qquad\qquad \triangleright$ Initialize upper bound of $\mathcal{P}_j$
9: $\quad \mathcal{V} \leftarrow V(\mathcal{U}_j)$ $\quad\triangleright$ Set of unchecked vertices of $\mathcal{U}_j$
10: $\quad$ **while** $\mathcal{V} \neq \varnothing$ **do**
11: $\qquad v \leftarrow$ Take any vertex in $\mathcal{V}$
12: $\qquad p \leftarrow \lambda p^{\text{int}} + (1-\lambda)v$
13: $\qquad a \leftarrow$ Oracle(p)
14: $\qquad$ **if** $a \neq a_j$ **then**
15: $\qquad\quad H_{jk} \leftarrow$ Find-Hyperplane$(a_j, \mathcal{U}_j, p^{\text{int}}, v, \delta)$
16: $\qquad\quad \mathcal{U}_j \leftarrow \mathcal{U}_j \cap \mathcal{H}_{jk}$ $\quad \triangleright$ Update upper bound
17: $\qquad\quad \mathcal{V} \leftarrow V(\mathcal{U}_j)$ $\quad \triangleright$ Update unchecked vertices
18: $\qquad$ **else**
19: $\qquad\quad \mathcal{V} \leftarrow \mathcal{V} \setminus \{v\}$ $\quad \triangleright$ Vertex v has been checked
20: $\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{a_j\}$ $\qquad \triangleright$ Action a_j has been closed
21: $p^\star \leftarrow \arg\max_{p \in \bigcup_{a_j \in \mathcal{C}} V(\mathcal{U}_j)} u_\ell(p)$

---

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).[4] 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 = $ Oracle(p^{int}) and $a_j \notin \mathcal{C}$. Then, the algorithm focuses on closing action a_j. First, it initializes the upper 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 = $ 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 4.2. 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

---

[4]Notice that $\bigcup_{a_j \in \mathcal{C}} \mathcal{U}_j \neq \Delta_m$ may *not* be convex. However, it can be expressed as union of a finite number of polytopes. As shown in Appendix E, this allows to apply the Sample-Int procedure in Section 4.4 to get the needed point in polynomial time, when either m or n is fixed.

bound (Line 17), and it starts checking vertices again. 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 1 is to check a vertex $v \in V(\mathcal{U}_j)$ by querying a nearby point in the interior of $\mathcal{U}_j$. This is obtained by moving towards the 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 4.1. *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 $\widetilde{p} := \lambda p + (1-\lambda)p'$ for some $\lambda \in (0, 2^{-m(B+4L)-1})$. Then, $\widetilde{p} \in \mathcal{P}_j \Leftrightarrow p' \in \mathcal{P}_j$.*

The two fundamental properties that hold when Algorithm 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 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 4.4.

Next, we show that the properties of Algorithm 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 21). 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.

Francesco Bacchiocchi, Matteo Bollini, Matteo Castiglioni, Alberto Marchesi, Nicola Gatti

---

Algorithm 2 Find-Hyperplane

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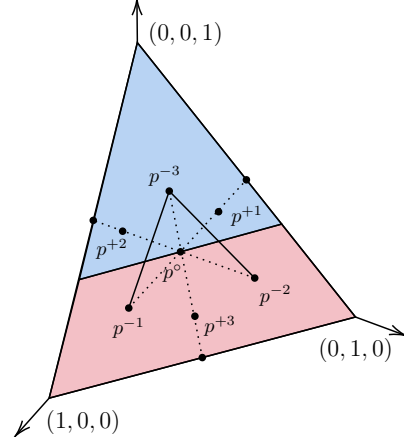

Figure 3: Example of points p^{+i} and p^{-i} computed by Algorithm 2, together with the $m-1$ line segments on which the algorithm performs binary search to compute the separating hyperplane H_{jk}.

Lemma 4.2. *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* $\mathcal{P}_j$.

Finally, by Lemmas 4.1 and 4.2, and all the results related to Find-Hyperplane and Sample-Int in Sections 4.2 and 4.4, respectively, we can bound the number of samples required by Algorithm 1.

Theorem 4.3. *Given any* $\zeta \in (0,1)$, *with probability at least* $1 - \zeta$, *Algorithm 1 terminates with* $p^\star$ *being an optimal strategy to commit to, by using at most* $\widetilde{\mathcal{O}}\left(n^2 \left(m^7 L \log(1/\zeta) + \binom{m+n}{m}\right)\right)$ *samples.*

By Theorem 4.3, the number of samples used by Algorithm 1 is polynomial when fixing either the number of leader's actions m or that of follower's actions n. Algorithm 1 requires an overall running time that is polynomial when either m or n is fixed (see Appendix E).

4.2 Find-Hyperplane

The Find-Hyperplane procedure is in Algorithm 2. Algorithm 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 $\texttt{Oracle}(v) \neq a_j$, and the parameter $\delta \in (0,1)$.

Algorithm 2 samples a point p in the interior of $\mathcal{U}_j$, using the Sample-Int procedure. 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 4.1 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 1).

After the two extremes p^1, p^2 of the segment have been identified, Algorithm 2 calls the Binary-Search procedure (Line 7). 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.[5] 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 11).[6] Then, it takes suitable linear combinations between sampled points and p°, defined by a parameter α (see Line 8). 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 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

---

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

[6]When $m = 2$, the loop at Line 10 can be skipped, as the algorithm can simply employ p° to compute H_{jk}.

response is a_k. Finally, Algorithm 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 shows an example of how Algorithm 2 works.

These are the fundamental properties of points p^{+i}:

Lemma 4.4. *With probability at least $1 - 2\delta m$, the points p^{+i} computed by Algorithm 2 belong to* $\mathrm{int}(\Delta_m)$, *are linearly independent, and are not on H_{jk}.*

Lemma 4.4 is proved by employing the properties of the procedure Sample-Int, shown in Lemma 4.8. The following lemma shows that the sets $\mathcal{S}_j$ and $\mathcal{S}_k$ built by Algorithm 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 4.5. *With probability at least $1 - \delta(m+n)^2$, any point p^{+i} or p^{-i} computed in Algorithm 2 is such that $a_f^\star(p^{+i}) \in \{a_j, a_k\}$. Moreover, if $p^{+i} \notin H_{jk}$ and $a_f^\star(p^{+i}) = a_j$, then $a_f^\star(p^{-i}) = a_k$, and viceversa.*

Lemma 4.5 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 4.5, with high probability, the sets $\mathcal{S}_j$ and $\mathcal{S}_k$ are well 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 4.4 and 4.5, we have:

Lemma 4.6. *With probability at least $1 - 2(m+n)^2\delta$, Algorithm 2 returns a separating hyperplane H_{jk} by using at most $\mathcal{O}(m^7 L + m^4 \log(1/\delta))$ samples.*

The number of samples can be bounded by observing that binary search is always performed between points with bit-complexity of the order of $\mathcal{O}(m^5 L + m^2 \log(1/\delta))$, and that $\mathcal{O}(m)$ binary searches are performed. This is ensured by Lemmas 4.8 and 4.7, and the fact that the bit-complexity of a vertex is $\mathcal{O}(m^2 L)$.

4.3 Binary-Search

The pseudocode of Binary-Search is provided in Algorithm 3. The binary search is performed on the line segment with extreme points p^1, p^2, with $\mathtt{Oracle}(p^1) = a_j$ and $\mathtt{Oracle}(p^2) \neq a_j$. 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

---

Algorithm 3 Binary-Search

Require: a_j, p^1, p^2 of bit-complexity $\leq B$ ▷ Algorithm 2
1: $\lambda_1 \leftarrow 0$; $\lambda_2 \leftarrow 1$
2: **while** $|\lambda_2 - \lambda_1| \geq 2^{-6m(5B+8L)}$ **do**
3:     $\lambda \leftarrow (\lambda_1 + \lambda_2)/2$; $p^\circ \leftarrow p^1 + \lambda(p^2 - p^1)$
4:     **if** $\mathtt{Oracle}(p^\circ) = a_j$ **then**
5:         $\lambda_1 \leftarrow \lambda$ ▷ New point with a_j as best response
6:     **else**
7:         $\lambda_2 \leftarrow \lambda$ ▷ New point with best response $\neq a_j$
8: $\lambda \leftarrow \mathtt{Stern\text{-}Brocot\text{-}Tree}(\lambda_1, \lambda_2, 3m(5B+8L))$
9: $p^\circ \leftarrow \lambda p^1 + (1-\lambda)p^2$

Algorithm 4 Sample-Int

Require: $\mathcal{P} \subseteq \Delta_m : \mathrm{vol}_{m-1}(\mathcal{P}) > 0 \vee \mathcal{P} := H_i \cap \Delta_m, \delta$
1: $\begin{cases} d \leftarrow m & \triangleright \text{ If } \mathcal{P} \subseteq \Delta_m : \mathrm{vol}_{m-1}(\mathcal{P}) > 0 \\ d \leftarrow m-1 & \triangleright \text{ If } \mathcal{P} = H_i \cap \Delta_m \end{cases}$
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: $x \sim \mathrm{Uniform}(\{-1, -\frac{M-1}{M}, \dots, 0, \dots, \frac{M-1}{M}, 1\}^{d-1})$
6: $l \leftarrow 1$; $k \leftarrow 1$
7: **while** $l \leq d - 1$ **do**
8:     **if** $k \neq i$ **then** $p_k \leftarrow p_k^\diamond + \rho x_l$; $l \leftarrow l + 1$
9:     **else** $p_k \leftarrow 0$;
10:     $k \leftarrow k + 1$
11: **if** $i = m$ **then** $p_{m-1} \leftarrow 1 - \sum_{l=1}^{m-2} p_l$
12: **else** $p_m \leftarrow 1 - \sum_{l=1}^{m-1} p_l$

---

extreme. The binary search goes on until 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 4.7. *Let $p^1, p^2 \in \Delta_m$ be two points such that (i) $a_f^\star(p^1) = a_j \neq a_f^\star(p^2)$ for $a_j \in A_f$, (ii) they are not both in $\mathcal{P}_j \setminus \mathrm{int}(\mathcal{P}_j)$, and (iii) they have bit-complexity bounded by B. Then, Algorithm 3 finds a point $p^\circ \in H_{jk}$ for some $a_k \in A_f$. Furthermore, the algorithm requires $\mathcal{O}(m(B + L))$ samples, and the point p° has bit-complexity upper bounded by $\mathcal{O}(m(B + L))$.*

4.4 Sample-Int

The pseudocode of Sample-Int is in Algorithm 4. It takes as input a polytope $\mathcal{P} \subseteq \Delta_m : \mathrm{vol}_{m-1}(\mathcal{P}) > 0$ or a facet $\mathcal{P} := H_i \cap \Delta_m$ and a probability $\delta \in (0, 1)$, and it returns a point from $\mathrm{int}(\mathcal{P})$. For the sake of exposition, suppose that $\mathcal{P}$ has volume larger than

zero. In this case, we consider $i = \texttt{null}$. First, Algorithm 4 computes $p^\diamond \in \text{int}(\mathcal{P})$ by taking the average of $d := m-1$ linearly-independent vertexes of $\mathcal{P}$ (Line 3). Then, the algorithm samples a point $x \in \mathbb{R}^{m-1}$ 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_k of x to a component $p_k^\diamond$ of $p^\diamond$ (while-loop at Line 7), by scaling it by a factor ρ (defined at Line 4). Observe that when $\text{vol}_{m-1}(\mathcal{P}) > 0$, $k = l$ during this loop. 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 4 belongs to $\text{int}(\mathcal{P})$.

The main property of Algorithm 4 is that, given any linear space $H \subseteq \mathbb{R}^m$ (with $\mathcal{P} \nsubseteq H$), 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, as they use strategies with arbitrarily large bit-complexity. The property is formalized as follows:

Lemma 4.8. *Given some polytope $\mathcal{P} \subseteq \Delta_m$: $\text{vol}_{m-1}(\mathcal{P}) > 0$ defined by separating or boundary hyperplanes, or a facet $\mathcal{P} := H_i \cap \Delta_m$, Algorithm 4 computes $p \in \text{int}(\mathcal{P})$ such that, for every linear space $H \subset \mathbb{R}^m : \mathcal{P} \nsubseteq H$ of dimension at most $m-1$, it holds $p \in H$ with probability at most δ. Furthermore, the bit-complexity of p is at most $40m^3L + 2\log_2(1/\delta)$.*

Lemma 4.8 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 the fact that the bit-complexity of all elements involved is suitably bounded.

Finally, let us describe how the execution of Algorithm 4 changes when $\mathcal{P} = H_i \cap \Delta_m$ is a facet of the simplex. In this case, we have $d = m-1$ and $x \in [-1,1]^{m-2}$. During the loop at Line 7, Algorithm 4 also sets $p_i = 0$, ensuring that $p \in H_i$. Thus, at the end of the loop, $m-2$ components of p have been set to some value. The remaining component is set so that the p_i sum to one.

Acknowledgments

References

Francesco Bacchiocchi, Matteo Castiglioni, Alberto Marchesi, and Nicola Gatti. Learning optimal contracts: How to exploit small action spaces. *arXiv preprint arXiv:2309.09801*, 2023.

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.

Dimitris Bertsimas and John Tsitsiklis. *Introduction to Linear Optimization*. Athena Scientific, 1st edition, 1997. ISBN 1886529191.

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.

Avrim Blum, Nika Haghtalab, MohammadTaghi Hajiaghayi, and Saeed Seddighin. Computing stackelberg equilibria of large general-sum games. page 168–182, 2019.

Matteo Castiglioni, Andrea Celli, Alberto Marchesi, and Nicola Gatti. Online bayesian persuasion. *Advances in Neural Information Processing Systems*, 33:16188–16198, 2020.

Alon Cohen, Argyrios Deligkas, and Moran Koren. Learning approximately optimal contracts. In *Algorithmic Game Theory: 15th International Symposium, SAGT 2022, Colchester, UK, September 12–15, 2022, Proceedings*, page 331–346, Berlin, Heidelberg, 2022. Springer-Verlag.

Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of*

the 7th ACM Conference on Electronic Commerce, 2006.

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*, pages 156–165, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-72914-3.

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.

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*, EC '14, page 359–376, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450325653.

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.

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.

Roger B Myerson. Optimal coordination mechanisms in generalized principal–agent problems. *Journal of mathematical economics*, 10(1):67–81, 1982.

Roger B Myerson. *Mechanism design*. Springer, 1989.

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.

Alexander Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, Inc., USA, 1986. ISBN 0471908541.

Heinrich Von Stackelberg. *Marktform und gleichgewicht*. J. springer, 1934.

Bernhard Von Stengel and Shmuel Zamir. Leadership games with convex strategy sets. *Games and Economic Behavior*, 69(2):446–457, 2010.

Banghua Zhu, Stephen Bates, Zhuoran Yang, Yixin Wang, Jiantao Jiao, and Michael I. Jordan. The sample complexity of online contract design. EC '23, page 1188, New York, NY, USA, 2023. Association for Computing Machinery.

You Zu, Krishnamurthy Iyer, and Haifeng Xu. Learning to persuade on the fly: Robustness against ignorance. In *EC*, pages 927–928, 2021.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

   (b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

   (c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [Not Applicable]

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. [Yes]

   (b) Complete proofs of all theoretical results. [Yes]

   (c) Clear explanations of any assumptions. [Yes]

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Not Applicable]

   (b) All the training details (e.g., data splits, hyperparameters, how they were chosen). [Not Applicable]

   (c) A clear definition of the specific measure or statistics and error bars (e.g., with respect to the random seed after running experiments multiple times). [Not Applicable]

(d) A description of the computing infrastructure used. (e.g., type of GPUs, internal cluster, or cloud provider). [Not Applicable]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Not Applicable]
   (b) The license information of the assets, if applicable. [Not Applicable]
   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]
   (d) Information about consent from data providers/curators. [Not Applicable]
   (e) Discussion of sensible content if applicable, e.g., personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]
   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]
   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

APPENDIX

The appendixes are organized as follows:

- Appendix A provides a detailed discussion of the previous works most related to ours.

- Appendix B provides a detailed discussion on instances in which the algorithmic approaches by Letchford et al. (2009) and Peng et al. (2019) fail.

- Appendix C provides all the proofs omitted from the main body of the paper.

- Appendix D discusses some useful technical details omitted from the paper.

- Appendix E provides a discussion on the running time of Algorithm 1.

- Appendix F provides a discussion on how the results presented in the main body of the paper generalize to the case in which there are equivalent follower's actions.

A   RELATED WORKS

Learning in Stackelberg games The two works closely related to ours are the ones by Letchford et al. (2009) and Peng et al. (2019), as discussed in Section 3. 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.

Learning in Stackelberg security games 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 strategy 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.

B   ISSUES OF THE ALGORITHMS BY PENG ET AL. (2019)
AND LETCHFORD ET AL. (2009)

B.1   Issue of the algorithm by Peng et al. (2019)

In the following, we present an instance in which the algorithm proposed by Peng et al. (2019) is not guaranteed to terminate correctly. Intuitively, the procedure proposed by Peng et al. (2019) maintains an upper bound for each best response region of the follower's action it has already discovered and proceeds by shrinking them by discovering new separating hyperplanes. One of the main differences with the approach by Letchford et al. (2009) is that, whenever the intersection of the upper bounds of the discovered follower's actions has zero measure, the algorithm enumerates the vertices of each upper bound and either terminates or discovers a new follower's action. Meanwhile, the algorithm also maintains a lower bound for the best response regions of the follower's action it has discovered so far.

We consider the instance presented in Figure 4, in which $|\mathcal{A}_\ell| = |\mathcal{A}_f| = 3$. As a first step, the algorithm proposed by Peng et al. (2019) prescribes to randomly sample a point from the interior of Δ_m, which coincides with the commitment $p \in \mathcal{P}_3$ in Figure 4. Then, it initializes the upper bound $\mathcal{U}_3$ with the whole simplex and

queries each vertex of such upper bound. When the algorithm queries the vertex v^1, it also chooses one of the two followers' best responses to eventually add 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, the algorithm updates the lower bound of the follower's action a_3, *i.e.*, $\mathcal{L}_3 = \mathrm{co}(v^2, v^3, p)$. 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 proposed by Letchford et al. (2009) and deterministically computes the point p^1 in Figure 4 lying at the intersection of the two upper bounds $\mathcal{U}_1$ and $\mathcal{U}_3 = \Delta_3$. Thus, following the steps presented in Letchford et al. (2009), the procedure 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 correctly terminates, it returns the vertex v^1 as the point ling on the hyperplane H_{12}. Consequently, at least one vertex of the small simplex centred in v^1 falls outside Δ_3 and cannot be queried, leading to a failure of the procedure by Peng et al. (2019).
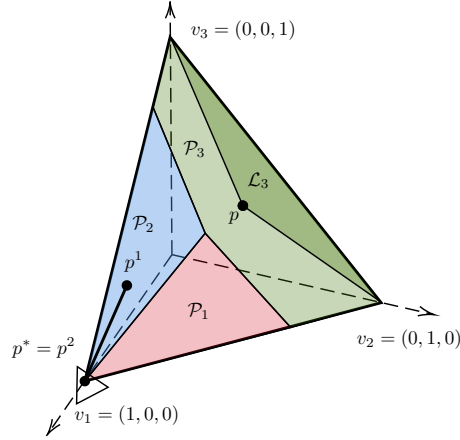

Figure 4: Instance in which the approach by Peng et al. (2019) fails due to how p^2 is selected

B.2 The algorithm by Peng et al. (2019) may require exponentially-many samples

In the following, we present an SG in which the binary search procedure proposed by by Letchford et al. (2009) and employed by the algorithm by Peng et al. (2019) to compute a separating hyperplane requires an exponential number of samples, if the leader's strategies are *not* specified with a bounded bit-complexity. We consider the scenario depicted in Figure 5. This refers to an SG in which both the leader and the follower have three actions available, namely $A_\ell = \{a_1, a_2, a_3\}$ and $A_f = \{a_1, a_2, a_3\}$. Furthermore, follower's utilities are specified by the following matrix:

$$u_f(a_i, a_j) = \begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

where the entry at position (i, j) encodes follower's payoff $u_f(a_i, a_j)$ when the leader plays action $a_i \in A_\ell$ and the follower plays action $a_j \in A_f$. The separating hyperplanes that define the three follower's best-response regions are given by:

$$\begin{cases} H_{12} := \{p \in \mathbb{R}^m \mid p_1 - p_2 = 0\}, \\ H_{13} := \{p \in \mathbb{R}^m \mid p_2 - p_3 = 0\}, \\ H_{23} := \{p \in \mathbb{R}^m \mid p_1 - p_3 = 0\}. \end{cases}$$

The algorithm by Peng et al. (2019) starts by sampling a point at random from the simplex that coincides with $p^1 \in \Delta_3$ in the example depicted in Figure 5. Then, the algorithm by Peng et al. (2019) observes the follower's best-response in p^1, which coincides with the follower's action a_1. Consequently, it initializes the relative upper bound $\mathcal{U}_1$ to the whole simplex Δ_m, and the lower bound $\mathcal{L}_1$ to the set $\{p^1\}$. As a further step, the algorithm checks the vertex $v^1 = (1, 0, 0)$ of $\mathcal{U}_1 = \Delta_m$. Since the follower's best-response in this point is a_2, it initializes the upper bound $\mathcal{U}_2$ to Δ_m and the the lower bound $\mathcal{L}_2$ to the set $\{v^1\}$. Subsequently, the algorithm by Peng et al. (2019) computes a hyperplane defining a facet of $\mathcal{P}_1$.
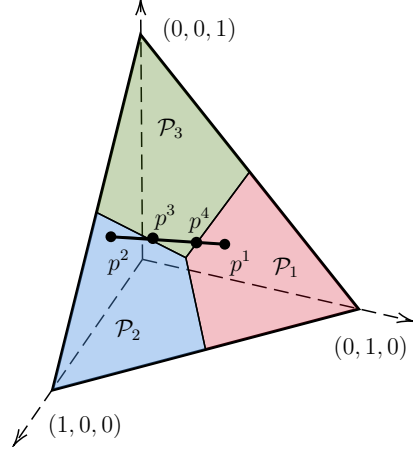
Figure 5: The binary search proposed employed by Peng et al. (2019) requires exponential samples.

To do so, it computes a point in the interior of intersection of the two upper bounds of the follower's best-response regions $\mathcal{P}_1$ and $\mathcal{P}_2$, which initially coincides with the whole simplex. If the sampled point coincides with p^2 in Figure 5, then the algorithm performs binary search between p^1 and p^2, since the follower's best response in p^2 coincides with a_2. Formally, the commitments p^1 and p^2 presented in Figure 5 are defined as $p^1 := (1/3 - \epsilon, 1/3 + \epsilon, 1/3) \in \mathcal{P}_1$ and $p^2 = (1/2, 1/10, 2/5) \in \mathcal{P}_2$, for some constant $\epsilon > 0$. Clearly, since the commitment p^1 is sampled at random from the simplex, a large number of bits may be needed, as ϵ could potentially be arbitrarily. We also define $p^3 \in \Delta_m$ as the point at the intersection of the line segment connecting p^1 to p^2 and the hyperplane H_{23}. Therefore, there exists a single value of $\lambda' \in (0, 1)$ such that $p^3 = \lambda'p^1 + (1 - \lambda')p^2$. Simple calculations show that the value of such a λ' is $(10\epsilon + 1)^{-1}$, which in turn implies that $p^3 = (30\epsilon + 3)^{-1}(12\epsilon + 1, 6\epsilon + 1, 12\epsilon + 1)$.

A crucial observation is that, to compute the point $p^4 \in \Delta_m$ at the intersection between the line segment that connects p^1 to p^2 and the hyperplane H_{13}, the binary search must employ at least $\mathcal{O}(\log_2(d_2(p^1,p^2)/d_2(p^1,p^3)))$ samples, where we let $d_2(p^1,p^2) := \|p^1 - p^2\|_2$ and $d_2(p^1,p^3) := \|p^1 - p^3\|_2$. This is because, to correctly identify the point $p^4 \in H_{13} \cap \text{co}(p^1,p^2)$, the binary search has to query at least one point that belongs to the line segment connecting p^1 to p^3, as $p^4 \in \text{co}(p^1,p^3)$. Thus, by observing that $p^1 - p^3 = 3(10\epsilon + 1)^{-1}(-30\epsilon^2 - 5\epsilon, 30\epsilon^2 + 7\epsilon, -2\epsilon)$, we can compute the two square distances $d_2(p^1,p^2)^2$ and $d_2(p^1,p^3)^2$ as follows.

$$d_2(p^1,p^2)^2 = \sum_{i \in [3]} (p_i^1 - p_i^2)^2 = \left(-\frac{1}{6} - \epsilon\right)^2 + \left(\frac{7}{30}\right)^2 + \left(\frac{1}{15}\right)^2 = 2\epsilon^2 + \frac{4}{5}\epsilon + \frac{13}{150}.$$

Similarly, the square distance $d_2(p^1,p^3)^2$ is equal to:

$$d_2(p^1,p^3)^2 = \sum_{i \in [3]} (p_i^1 - p_i^3)^2$$
$$= \frac{1}{9(10\epsilon + 1)^2}\left((30\epsilon^2 + 5\epsilon)^2 + (30\epsilon^2 + 7\epsilon) + 4\epsilon^2\right)$$
$$= \frac{1800\epsilon^4 + 720\epsilon^3 + 78\epsilon^2}{9(100\epsilon^2 + 20\epsilon + 1)}$$

As a result, the ratio between such distances is given by:

$$\frac{d_2(p^1,p^2)}{d_2(p^1,p^3)} = \sqrt{\frac{9(2\epsilon^2 + \frac{4}{5}\epsilon + \frac{13}{150})(100\epsilon^2 + 20\epsilon + 1)}{1800\epsilon^4 + 720\epsilon^3 + 78\epsilon^2}} = \mathcal{O}\left(\frac{1}{\epsilon}\right).$$

As previously observed, the number of samples required by the binary search to be correctly executed is of the order $\mathcal{O}(\log_2(d_2(p^1,p^2)/d_2(p^1,p^3)))$ and, consequently, $\mathcal{O}(\log_2(1/\epsilon))$. Therefore, when $\epsilon = 2^{-2^L}$, the sample complexity of such a procedure is $\mathcal{O}(2^L)$, showing an exponential dependence in the parameter L. Intuitively,

this is because the point p^1 is taken arbitrarily close to the commitment $p = (1/3, 1/3, 1/3)$, which lies at the intersection of the three best response regions. As a consequence, it is necessary to bound the bit complexity of the leader's commitments to avoid an exponential dependence on the number of samples required to compute an optimal solution.[7]

B.3   Minor issue in the procedure by Letchford et al. (2009)

In this section, we present an instance in which the algorithm proposed by Letchford et al. (2009) and employed by Peng et al. (2019) to compute a separating hyperplane does not terminate correctly. Given two overestimates $\mathcal{U}_j$ and $\mathcal{U}_k$ of some best-response regions $\mathcal{P}_j$ and $\mathcal{P}_k$ such that $\mathrm{vol}(\mathcal{U}_j \cap \mathcal{U}_k) > 0$, this procedure finds a separating hyperplane H_{jh} for some $h \in [h]$. Observe that $\mathcal{P}_j$ and $\mathcal{P}_k$ may not share a facet, and thus a_h can be different from a_k. To identify this hyperplane, the algorithm deterministically computes a point $p^1 \in \mathrm{int}(\mathcal{U}_j \cap \mathcal{U}_k)$. Subsequently, it performs a binary search on the segment connecting p^1 to a suitable point p^2, where p^2 belongs to either $\mathcal{L}_j$ or $\mathcal{L}_k$ depending on the follower's best response in p^1. Then, a random simplex of dimension m is built centered at the point returned by the former procedure, and new binary searches are performed between the vertices of such a simplex in which different actions are implemented. In this way, the algorithm eventually computes a set of linearly independent points lying on the hyperplane, which is enough to uniquely identify it.
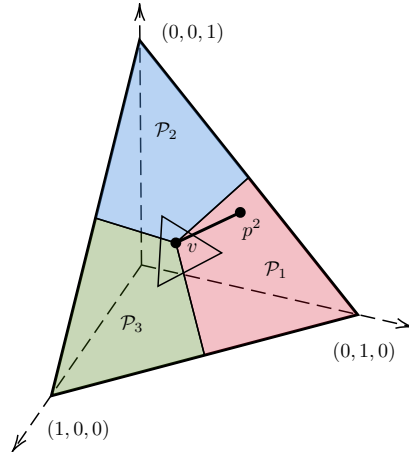

Figure 6: Instance in which the procedure by Letchford et al. (2009) to compute hyperplanes fails

The issue with the algorithm proposed by Letchford et al. (2009) lies in the choice of the segment where the binary search is executed. Specifically, we observe that the point p^1 should *not* be chosen deterministically. In the following, we consider an instance (see Figure 6) where $|\mathcal{A}_\ell| = |\mathcal{A}_f| = 3$. Furthermore, we assume that the three follower's best-response regions share a vertex, *i.e.*, $\mathcal{P}_1 \cap \mathcal{P}_2 \cap \mathcal{P}_3 = v$ and the action actually played by the follower in such a commitment is equal to a_3, namely $a_f^\star(v) = a_3$. The algorithm by Peng et al. (2019) samples a random point from the simplex, which we suppose to be the point p^2 in Figure 6. Since $a_f^\star(p^2) = a_1$, the algorithm initializes the upper bound $\mathcal{U}_1$ of $\mathcal{P}_1$ to the simplex Δ_3, and the corresponding lower bound $\mathcal{L}_1$ to $\{p^1\}$. Subsequently, it queries the vertex $v^1 = (1, 0, 0)$ of $\mathcal{U}_1$. Thus, it discovers the follower's action a_3 and initializes $\mathcal{U}_3 = \Delta_3$ and $\mathcal{L}_3 = \{a_3\}$.

At this point, the Algorithm by Peng et al. (2019) employs the procedure proposed by Letchford et al. (2009) to compute a new separating hyperplane. In particular, it computes deterministically a point p^1 in the interior of $\mathcal{U}_1 \cap \mathcal{U}_3 = \Delta_3$. If such a point is the vertex v at the intersection of the three best-response regions, then the algorithm performs a binary search on the segment connecting $p^1 = v$ and $p^2 \in \mathcal{L}_1$, as $a_f^\star(v) = a_3 \neq a_1$. Thus, the point v coincides with the commitment returned by such a procedure, which belongs to the hyperplane H_{13}. As a further step, the procedure by Letchford et al. (2009) attempts to organize the vertices of a small simplex centered in v into two sets in which either a_1 or a_3 is a best response. However, such a procedure fails no matter the how small the simplex is drawn, as its three vertices lie in three (and not two) different best-response regions. This example shows that if the point p^1 over which the binary search is performed is chosen deterministically, there may be instances in which the procedure proposed by Letchford et al. (2009) fails with probability one.

---

[7]Given $n \in \mathbb{N}_+$, we let $[n] := \{1, \ldots, n\}$.

C   OMITTED PROOFS

C.1   Proofs Omitted from Learn-Optimal-Commitment

Lemma 4.1. *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 $\widetilde{p} := \lambda p + (1 - \lambda)p'$ 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 immediately 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 $p^\circ := \text{co}(p, p') \cap \mathcal{H}_{kj}$. Intuitively, p° coincides with the intersection between the separating hyperplane $\mathcal{H}_{kj}$ 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 $p' \in \mathcal{H}_{kj}$, 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 D.1 in Appendix D. 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:

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

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. $\qquad \square$

Lemma 4.2. *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 $\text{vol}(\mathcal{P}_j) > 0$. 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 $\mathcal{H}_{jk}$, since $\{a_k, a_j\} \subseteq A(p^\star)$. As a result, $p^\star \in \mathcal{P}_j \cap \mathcal{H}_{jk}$. Moreover, we have that $\mathcal{P}_j \cap \mathcal{H}_{jk} \subseteq \mathcal{P}_k$. This holds since in any point in the set $\mathcal{P}_j \cap \mathcal{H}_{jk}$ 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 $\mathcal{H}_{jk}$.

Finally, it is easy to see that $V(\mathcal{P}_j \cap \mathcal{H}_{jk}) \subseteq V(\mathcal{P}_j)$ by the definition of $\mathcal{P}_j$. By the convexity of $\mathcal{P}_j \cap \mathcal{H}_{jk}$ and the fact that leader's expected utility is linear over $\mathcal{P}_j \cap \mathcal{H}_{jk}$ 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.

$\square$

Theorem 4.3. *Given any $\zeta \in (0,1)$, with probability at least $1 - \zeta$, Algorithm 1 terminates with $p^\star$ being an optimal strategy to commit to, by using at most $\widetilde{\mathcal{O}}\left(n^2\left(m^7 L \log(1/\zeta) + \binom{m+n}{m}\right)\right)$ samples.*

Proof. First, we prove that, with probability at least $1 - \delta n^2(2(m+n)^2 + n)$, Algorithm 1 returns an optimal commitment in a finite number of steps. To do this, we employ an inductive argument on the different executions of the while loop at Line 10. Formally, we show that, if before the execution of the while loop at Line 10 in Algorithm 1 it holds $\text{vol}(\mathcal{P}_j) > 0$ and $\mathcal{U}_j = \mathcal{P}_j$ for every action $a_j \in \mathcal{C}$, then such a while loop terminates in a finite number of rounds. Furthermore, the set $\mathcal{C}$ is updated as $\mathcal{C} \leftarrow \mathcal{C} \cup \{a_k\}$, where $\text{vol}(\mathcal{P}_k) > 0$ and $\mathcal{U}_k = \mathcal{P}_k$, with probability at least of $1 - \delta n(2(m+n)^2 + n)$.

We start by noticing that, with probability at least $1 - \delta n^2$, the new action $a_k \in A_f$ added to $\mathcal{C}$ is such that $\text{vol}(\mathcal{P}_k) > 0$. This is because the commitment p^{int} is sampled outside the union of the best-response regions $\mathcal{P}_j$ with $a_j \in \mathcal{C}$. Indeed, for every action $a_j \in \mathcal{C}$, it holds that $\mathcal{P}_j = \mathcal{U}_j$, thanks to the inductive hypothesis. As a result, $a_f^\star(p^{\text{int}}) \neq a_j$ for every $a_j \in \mathcal{C}$. Furthermore, with probability at least $1 - \delta n^2$, the strategy p^{int} does not belong to any separating hyperplane, as guaranteed by Lemma 4.8 and a union bound on the total number of separating hyperplanes, which is at most n^2. Consequently, $p^{\text{int}} \in \text{int}(\mathcal{P}_k)$ for some $a_k \in A_f$, proving that $\text{vol}(\mathcal{P}_k) > 0$.

As a further step, we prove that if $p^{\text{int}} \in \text{int}(\mathcal{P}_k)$ with $\text{vol}(\mathcal{P}_k) > 0$, then with probability at least $1 - 2\delta n(m+n)^2$, the while loop at Line 10 terminates in a finite number of rounds. This is because, with probability at least $1 - 2\delta n(m+n)^2$, during the execution of such a loop, the algorithm never computes the same separating hyperplane H_{kl} with $a_l \in A_\ell$ multiple times. Clearly, this hold only in the case, whenever invoked, Algorithm 2 correctly identifies the separating hyperplane. Observe that during the execution of the while loop at Line 10, Algorithm 2 is always invoked receiving as input a commitment $p^{\text{int}} \in \text{int}(\mathcal{P}_k)$ and a vertex v in which $a_f^\star(v) \neq a_j$, as ensured by Lemma 4.1. Furthermore, the point p° found by binary search at Line 7 in Algorithm 2 belongs to both a separating hyperplane H_{kl} and the interior of $\mathcal{U}_k$, thus it does not belong to a previously discovered hyperplanes defining $\mathcal{U}_k$. As a result, Algorithm 2 returns a new hyperplane with probability at least $1 - 2\delta(m+n)^2$ thanks to Lemma 4.6. Furthermore, by observing that Algorithm 2 is invoked at most n different times and each time it returns a new hyperplane with a probability of $1 - 2\delta(m+n)^2$, we have that the while loop at Line 10 terminates in a finite number of steps with probability at least $1 - 2\delta n(m+n)^2$, by employing a union bound over all the possible calls to Algorithm 2.

Moreover, we show that, if during the execution of the while loop at Line 10 Algorithm 2 correctly identifies a new separating hyperplane for the region $\mathcal{P}_k$ whenever invoked, then $\mathcal{U}_k = \mathcal{P}_k$. This is because, throughout the execution of the while loop at Line 10, it always holds that $\mathcal{P}_k \subseteq \mathcal{U}_k$, by how Algorithm 1 works. At the same time, when such a loop terminates, we have $\mathcal{U}_k = \text{co}(V(\mathcal{U}_k))$ with $a_f^\star(p) = a_k$ for each $p \in V(\mathcal{U}_k)$, and thus $\mathcal{U}_k \subseteq \mathcal{P}_k$, proving that $\mathcal{P}_k = \mathcal{U}_k$.

As a result, after the execution of the while loop at Line 10, by combining the above observations, we have that the set $\mathcal{C}$ is updated as $\mathcal{C} \leftarrow \mathcal{C} \cup \{a_k\}$, where $\text{vol}(\mathcal{P}_{a_k}) > 0$, $\mathcal{U}_k = \mathcal{P}_k$, and $a_k \notin \mathcal{C}$ with probability at least $1 - \delta n(2(m+n)^2 + n)$. Moreover, we observe that the initial step of the induction, *i.e.*, when $\mathcal{C} = \varnothing$, can be proved with the same argument as above. Therefore, we conclude that when Algorithm 1 terminates, the union of the sets $\mathcal{U}_j$ with $a_j \in \mathcal{C}$ coincides with the simplex Δ_m with probability at least $1 - \delta n^2(2(m+n)^2 + n)$. Such a result follows by induction and employing a union bound over all the possible executions of the while loop at Line 10, which are at most n. Therefore, by means of Lemma 4.2, we have that Algorithm 1 returns an optimal strategy to commit to in a finite number of steps wit probability at least $1 - \delta n^2(2(m+n)^2 + n)$.

Finally, we observe that, with probability at least $1 - \delta n^2(2(m+n)^2 + n)$, the number of samples required by Algorithm 1 is equal to $\mathcal{O}\left(n^2\left(m^7 L \log\left(\frac{1}{\delta}\right) + \binom{m+n}{m}\right)\right)$. This follows from the observation that during the execution of the loop at Line 10, Algorithm 2 is invoked at most n times, and the number of samples required by such a procedure is $\mathcal{O}\left(m^7 L \log\left(\frac{1}{\delta}\right)\right)$, as guaranteed by Lemma 4.6. Additionally, before computing an hyperplane, in the worst-case scenario Algorithm 1 has queried every vertex of the region $\mathcal{U}_k$. Since the number of vertexes of each $\mathcal{U}_k$ is bounded by $\binom{m+n}{m}$, and such a check is performed at most n times during the execution of the loop, the

number of samples required to execute the while loop at Line 10 is $\mathcal{O}\left(n\left(m^7 L \log\left(\frac{1}{\delta}\right) + \binom{m+n}{m}\right)\right)$. Consequently, observing that the number of follower's actions is n, the while loop at Line 10 is executed at most n times. Thus, the overall number of samples required by Algorithm 1 is of the order of $\mathcal{O}\left(n^2\left(m^7 L \log\left(\frac{1}{\delta}\right) + \binom{m+n}{m}\right)\right)$. As a result, since $\zeta = \delta n^2(2(m+n)^2 + n)$, with probability at least $1 - \zeta$, the number of samples required by Algorithm 1 is $\widetilde{\mathcal{O}}\left(n^2\left(m^7 L \log\left(1/\zeta\right) + \binom{m+n}{m}\right)\right)$, which concludes the proof. $\qquad\square$

C.2 Omitted Proofs from Separating-Hyperplane

Lemma 4.4. *With probability at least $1 - 2\delta m$, the points p^{+i} computed by Algorithm 2 belong to $\mathrm{int}(\Delta_m)$, are linearly independent, and are not on H_{jk}.*

Proof. First, we observe that $p^\circ \in \mathrm{int}(\Delta_m)$. This is because the binary search is always performed between an interior point $p \in \mathrm{int}(\Delta_m)$, in which $a_f^\star(p) = a_j$, and another point that is either an interior point or a vertex v in which follower's best response does not coincide with a_j (as guaranteed by Algorithm 1 and Lemma 4.1). Thus, the resulting p° cannot coincide with the vertex v itself. Next, we show that each p^{+i} with $i \in [m]$ belongs to $\mathrm{int}(\Delta_m)$. To prove that, notice that:

$$\sum_{i\in[m]} p_i^{+i} = \sum_{i\in[m]} p_i^\circ + \sum_{i\in[m]} \alpha(p_i - p_i^\circ) = 1$$

and:

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

for every $j \in [m]$. Notice that the inequality above holds because p_j° is at least 2^{-B} for every $j \in [m]$, given that the bit-complexity of p° is bounded by B, and the fact that p° belongs to the interior of Δ_m. Therefore, noticing that $\alpha(p_j - p_j^\circ) \geq -\alpha$ for every $j \in [m]$, it follows that each p^{+i} belongs to Δ_m, with each component strictly greater than zero. Thus, this proves that all the points p^{+i} belong to $\mathrm{int}(\Delta_m)$.

As a further step, we prove that, with probability at least $1 - \delta m$, all the points p^{+i} with $i \in [m]$ computed by Algorithm 2 are linearly independent. To do that, at each iteration $k \in [m]$ of the for loop at Line 10, we define $H_k' := \mathrm{span}\left\{p^{+1}, \ldots, p^{+k}\right\}$ as the linear space generated by the linear combinations of the points p^{+k} computed up to round $k \in [m]$. To ensure that all these commitment are linearly independent, we have to guarantee that $p^{+(k+1)} \notin H_k'$ for every $k \in [m-1]$. This condition is true only if the point p sampled from the facet $H_{k+1} \cap \Delta_m$ does not belong to the linear space defined as

$$H_k'' := \left\{\sum_{l=1}^{k} \lambda_l p^{+l} + \left(1 - 1/\alpha\right) p^\circ \mid \lambda_l \in \mathbb{R} \quad \forall l \in [k]\right\}.$$

This is because, with a simple calculation, we can show that $p^{+(k+1)}$ belongs to H_k' if and only if p sampled from $H_{k+1} \cap \Delta_m$ belongs to H_k''. As a result, we have to bound the probability that $p \notin H_k'' \cap (H_{k+1} \cap \Delta_m)$ for every $k \in [m-1]$. We notice that, when $k \leq m-2$, the linear space H_k'' has dimension at most $m-2$ and does not coincide with $H_{k+1} \cap \Delta_m$, thus $H_k'' \cap (H_{k+1} \cap \Delta_m)$ has dimension at most $m-3$. When $k = m-1$, we observe that p sampled from $\mathrm{int}(H_1 \cap \Delta_m)$ is such that $p_i > 0$ for every $i \geq 2$, and thus $p \notin H_m$. As a result, $H_{m-1}'' \cap \Delta_m \nsubseteq H_m$. Consequently, $H_k'' \cap (H_{k+1} \cap \Delta_m)$ has a dimension of at most $m-3$. Thus, it holds that $H_{k+1} \cap \Delta_m \nsubseteq H_k''$ for every $k \in [m-1]$. Consequently, Algorithm 4 ensures that the probability of sampling $p \in H_k'' \cap (H_{k+1} \cap \Delta_m)$ is at most δ, for every $k \in [m-1]$. Therefore, by employing a union bound, all the points p^{+k} with $k \in [m]$ are linearly independent with probability at least $1 - \delta m$

To conclude the proof, we bound the probability that each p^{+i} with $i \in [m]$ does not belong to the separating hyperplane H_{jk}. We observe that p^{+i} belongs to $H_{jk} \cap \Delta_m$ only when the point p sampled from $H_i \cap \Delta_m$ belongs to H_{jk}. This is because p^{+i} is on the line segment connecting p to p°, and such a segment belongs to H_{jk} only if p also belongs to H_{jk}. Additionally, we notice that, for every $i \in [m]$, the linear space $H_{jk} \cap (H_i \cap \Delta_m)$ has dimension $m-2$ only in the case in which $H_i = H_{ij}$. Such a scenario is not possible since $p^\circ \in \mathrm{int}(\Delta_m) \cap H_{ij}$, and thus $p_i^\circ \neq 0$, which in turn implies that $p^\circ \notin H_i$ for every $i \in [m]$. Therefore, the dimension of $H_{jk} \cap (H_i \cap \Delta_m)$ is at most $m-3$. As a result, Algorithm 4 ensures that the probability of sampling $p \in H_{jk} \cap (H_i \cap \Delta_m)$ is at most δ, since $(H_i \cap \Delta_m) \nsubseteq H_{jk}$ for every $i \in [m]$. As a result, by employing a union bound over all the points sampled from the m facets of the simplex, we have 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 $1 - 2\delta m$. $\qquad\square$

Lemma 4.5. *With probability at least $1 - \delta(m + n)^2$, any point p^{+i} or p^{-i} computed in Algorithm 2 is such that $a_f^\star(p^{+i}) \in \{a_j, a_k\}$. Moreover, if $p^{+i} \notin H_{jk}$ and $a_f^\star(p^{+i}) = a_j$, then $a_f^\star(p^{-i}) = a_k$, and viceversa.*

Proof. As a first step, we bound the probability that the point p° computed at Line 7 in Algorithm 2 belongs to a *single* hyperplane H_{jk} with $k \in [n]$ or to multiple coinciding hyperplanes H_{jk} for $k \in [n]$, and thus, p° belongs to the interior of a facet of $\mathcal{P}_j$. Let H, H' be either separating hyperplanes or boundary planes defining two different facets of $\mathcal{P}_j \subseteq \Delta_m$. In the following, we bound the probability that the line segment connecting p^1 to p^2 does not intersect any $H \cap H'$ with non-empty intersection. To prove that, we define $\mathcal{P} = H \cap H'$ as the linear space defined as the non-empty intersection of H and H'. We observe that such a linear space has a dimension of $m - 2$, given that H and H' are distinct and non-parallel hyperplanes corresponding to different facets of $\mathcal{P}_j$. Consequently, there exists a linear space H'', defined as the linear combination of the point p^2 where the binary search is conducted and the linear space $\mathcal{P}$, formally defined as $H'' = \mathrm{span}\{p^2, \mathcal{P}\}$, with a dimension at most $m - 1$. To prove the statement of the lemma, we first consider the case in which the randomly sampled point p^1 is such that $a_f^\star(p^1) = a_j$. Then, in order to ensure that the binary search determines a point p° that belongs to a single or possibly coinciding hyperplanes H_{jk}, we have to bound the probability that $p^1 \notin H''$. We notice that such a probability is greater or equal to $1 - \delta(m + n)^2$. Such a result follows by means of both Lemma 4.8 and a union over all the possible pairs of hyperplanes H and H' defining the facets of $\mathcal{P}_j$, which are at most $\binom{m+n}{2} \le (m + n)^2$. Furthermore, if $a_f^\star(p^1) \ne a_j$ and $p^2 = p^{\mathrm{int}}$, we can employ the same argument to bound the probability that p^1 does not belong to H''. Thus, by combining the two previous results and employing the law of total probability, the point p° lies on a single or possibly coinciding separating hyperplanes H_{jk} with $k \in [n]$ with probability of at least $1 - \delta(m + n)^2$.

We also define $\mathbb{S}_\epsilon^m(p^\circ) := \{x \in \mathbb{R}^m \mid \|x - p^\circ\|_2 \le \epsilon\}$ 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:

$$d_2(p^\circ, \widetilde{H}_{lm}) = \left| \frac{\sum_{i=1}^m p_i^\circ \left(u^f(a_i, a_l) - u^f(a_i, a_m) \right)}{\sqrt{\sum_{i=1}^m \left(u^f(a_i, a_l) - u^f(a_i, a_m) \right)^2}} \right| \ge \frac{1}{m 2^{m(B+4L)}}$$

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 D.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 $\mathcal{P}_k$. Consequently, since p° belongs to a single hyperplane H_{jk} with $\mathrm{vol}(\mathcal{P}_k) > 0$, we have that the sphere $\mathbb{S}_\epsilon^m(p^\circ) \cap \Delta_m$ belongs to $\mathrm{int}(\mathcal{P}_j \cup \mathcal{P}_k)$.

We also show that all the commitments p^{+i} with $i \in [m]$ computed in Algorithms 2 are such that $p^{+i} \in \mathbb{S}_\epsilon^m(p^\circ)$ with $\epsilon < \epsilon' = 2^{-m(B+2L)}/m$. Formally, we have:

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

The first equality holds because of the definition of p^{+i}, while the second inequality holds because $\max_{p,p' \in \Delta_m} \|p - p'\|_2 \le \sqrt{2}$ and thanks to the definition of $\alpha = 2^{-m(B+4L)-1}/m$. Thus, observing that each $p^{+i} \in \mathbb{S}_\epsilon^m$ for each $i \in [m]$, we have that in each p^{+i} either a_j or a_k is a 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:

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

This, in turn, implies that:

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

since $p^\circ \in H_{jk}$. 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. $\qquad \square$

Lemma 4.6. *With probability at least $1 - 2(m+n)^2\delta$, Algorithm 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 4.4. Furthermore, employing Lemma 4.5, we have that in each point p^{+i} either a_j or a_k is a follower's best response. As a result, employing a union bound, we can compute by binary search $m - 1$ points lying on the separating hyperplane H_{jk} with probability at least $1 - 2\delta(m+n)^2$.

To conclude the proof, we observe that the binary search to compute p° is consistently performed between a point sampled according to Algorithm 4, whose bit-complexity is bounded by $\mathcal{O}(Lm^3 + \log(\frac{1}{\delta}))$ as a consequence of Lemma 4.8, and either another interior point or a vertex. Therefore, since the bit-complexity of each vertex can be bounded by $O(m^2 L)$ by means of Lemma D.2, the number of rounds required to compute p° by means of Algorithm 3 is $\mathcal{O}(m^4 L + m \log(\frac{1}{\delta}))$ as prescribed by Lemma 4.7. Furthermore, the bit-complexity of p° is bounded by $\mathcal{O}(m^4 L + m \log(\frac{1}{\delta}))$. As a result, the bit-complexity of the parameter α is equal to $\mathcal{O}(m^5 L + m^2 \log(\frac{1}{\delta}))$. 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 $\mathcal{O}(m^5 L + m^2 \log(\frac{1}{\delta}))$. Consequently, each of the final $m - 1$ binary searches performed to compute the $m - 1$ points on the separating hyperplane requires at most $\mathcal{O}(m^6 L + m^3 \log(\frac{1}{\delta}))$ samples and returns a points on the hyperplane with bit-complexity bounded by $\mathcal{O}(m^6 L + m^3 \log(\frac{1}{\delta}))$. Therefore, Algorithm 2 requires $\mathcal{O}(m^7 L + m^4 \log(\frac{1}{\delta}))$ samples by accounting for all the $m - 1$ binary search performed. $\qquad \square$

C.3 Proofs Omitted from Binary-Search

Lemma 4.7. *Let $p^1, p^2 \in \Delta_m$ be two points such that (i) $a_f^\star(p^1) = a_j \neq a_f^\star(p^2)$ for $a_j \in A_f$, (ii) they are not both in $\mathcal{P}_j \setminus \text{int}(\mathcal{P}_j)$, and (iii) they have bit-complexity bounded by B. Then, Algorithm 3 finds a point $p^\circ \in H_{jk}$ for some $a_k \in A_f$. Furthermore, the algorithm requires $\mathcal{O}(m(B + L))$ samples, and the point p° has bit-complexity upper bounded by $\mathcal{O}(m(B + L))$.*

Proof. We notice that the point p° returned by Algorithm 3 lies on line segment that connect p^1 to p^2. This implies that $p^\circ \leftarrow p^1 + \lambda^\circ(p^2 - p^1)$ for some $\lambda^\circ \in [0, 1]$. Furthermore, p° also lies on a separating hyperplane H_{jk}, thus the following holds $\sum_{i \in [m]} p_i^\circ(u_f(a_i, a_j) - u_f(a_i, a_k)) = 0$. Then, by combining the two equations we get:

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

We observe that the bit complexity of the numerator defining λ° is the sum of m terms with a bit complexity bounded by $B + 4L$. This is because the terms encoding the difference in the follower's utility, as prescribed by Lemma D.1, have a bit complexity bounded by $4L$. Therefore, the sum of these m terms has a bit complexity of $3m(B + 4L)$ due to Lemma D.1. Similarly, we can show that the denominator that defines λ° has a bit complexity bounded by $3m(4B + 4L)$. As a result, the bit complexity of λ° is bounded by $3m(5B + 8L)$, which is of the order $\mathcal{O}(m(B + L))$.

Moreover, the minimum distance between two valid values of λ° is lower-bounded by $\epsilon := 2^{-6m(5B+8L)}$, according to Lemma D.1. Therefore, when Algorithm 3 terminates, there exists a unique λ° within the interval $[\lambda_1, \lambda_2]$, with bit complexity bounded by $3m(5B + 8L)$, given that the distance between λ_1 and λ_2 satisfies $|\lambda_1 - \lambda_2| \leq \epsilon$. Consequently, the bit complexity of the resulting point p° lying on the separating hyperplane H_{jk} is bounded

by $\mathcal{O}(m(B + L))$. This is because $p^\circ \leftarrow p^1 + \lambda^\circ(p^2 - p^1)$ is defined as the sum of two rational numbers with bit complexities bounded by $3m(5B + 8L) + B$, and thus it is bounded by $24m(3B + 4L)$. Lastly, we observe that the number of rounds required by Algorithm 3 to terminate is on the order of $\mathcal{O}(\log_2(1/\epsilon))$, and thus $\mathcal{O}(m(B + L))$. $\square$

C.4 Proofs Omitted from Sample-Point

Lemma 4.8. *Given some polytope $\mathcal{P} \subseteq \Delta_m : \mathrm{vol}_{m-1}(\mathcal{P}) > 0$ defined by separating or boundary hyperplanes, or a facet $\mathcal{P} := H_i \cap \Delta_m$, Algorithm 4 computes $p \in \mathrm{int}(\mathcal{P})$ such that, for every linear space $H \subset \mathbb{R}^m : \mathcal{P} \not\subseteq H$ of dimension at most $m - 1$, it holds $p \in H$ with probability at most δ. Furthermore, the bit-complexity of p is at most $40m^3 L + 2\log_2(1/\delta)$.*

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 $\mathrm{vol}_{m-1}(\mathcal{P}) > 0$, the case when $\mathcal{P} = H_i \cap \Delta_m$ can be proved with the same analysis. In the following, we let $H := \{x \in \mathbb{R}^m \mid \sum_{i=1}^m \alpha_i x_i = \beta\}$, be an hyperplane in $\mathbb{R}^m$. We show that the probability that the commitment p returned by Algorithm 4 belongs to H is equal to the probability that $x \in \Xi := \{-1, -\frac{M-1}{M}, \ldots, 0, \ldots, \frac{M-1}{M}, 1\}^{m-1}$, sampled according to a uniform probability distribution over the set Ξ, belongs to a suitably defined linear space H'.

$$
\begin{aligned}
\mathbb{P}(p \in H) &= \mathbb{P}\left(\sum_{i=1}^{m-1} \alpha_i p_i + \alpha_m \left(1 - \sum_{i=1}^{m-1} p_i\right) = \beta\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, p_i = \beta - \alpha_m\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, (p_i^\diamond + \rho x_i) = \beta - \alpha_m\right) \\
&= \mathbb{P}\left(\rho \sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, x_i = \beta - \alpha_m - \sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, p_i^\diamond\right) \\
&= \mathbb{P}\left(\sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, x_i = \frac{1}{\rho}\left(\beta - \alpha_m - \sum_{i=1}^{m-1} (\alpha_i - \alpha_m) \, p_i^\diamond\right)\right) \\
&= \mathbb{P}(x \in H'),
\end{aligned}
$$

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 probability over the set Ξ can be written as follows:

$$
\mathbb{P}_{x \sim \mathcal{U}(\Xi)}(x = \bar{x}) = \frac{1}{\mathrm{vol}(Q)} \int_Q \mathbb{1}\left\{\|s - \bar{x}\|_\infty \leq \frac{1}{2M}\right\} ds
$$

for each $\bar{x} \in \Xi$, where $Q := \{x \in \mathbb{R}^{m-1} \mid |x_i| \leq 1 + 1/2M \; \forall i \in [m-1]\}$ is an hypercube in dimension $m - 1$. Then, the following holds:

$$
\begin{aligned}
\mathbb{P}_{x \sim \mathcal{U}(\Xi)}(x \in H') &= \sum_{\bar{x} \in \Xi} \mathbb{1}\{\bar{x} \in H'\} \, \mathbb{P}_{x \sim \mathcal{U}(\Xi)}(x = \bar{x}) \\
&= \frac{1}{\mathrm{vol}(Q)} \sum_{\bar{x} \in \Xi} \mathbb{1}\{\bar{x} \in H'\} \int_Q \mathbb{1}\left\{\|s - \bar{x}\|_\infty \leq \frac{1}{2M}\right\} ds \\
&= \frac{1}{\mathrm{vol}(Q)} \sum_{\bar{x} \in \Xi} \int_Q \mathbb{1}\left\{s \in Q \mid \|s - \bar{x}\|_\infty \leq \frac{1}{2M}, \bar{x} \in H'\right\} ds \\
&= \frac{1}{\mathrm{vol}(Q)} \int_Q \mathbb{1}\left\{s \in Q \mid \|s - \bar{x}\|_\infty \leq \frac{1}{2M}, \bar{x} \in H' \cap \Xi\right\} ds \\
&\leq \frac{1}{\mathrm{vol}(Q)} \int_Q \mathbb{1}\left\{s \in Q \mid \|s - t\|_\infty \leq \frac{1}{2M}, t \in H'\right\} ds
\end{aligned}
$$

$$= \frac{1}{\text{vol}(Q)} \int_Q \mathbb{1}\left\{ d_\infty(s, H') \leq \frac{1}{2M} \right\} ds$$

$$\leq \frac{1}{\text{vol}(Q)} \int_Q \mathbb{1}\left\{ d_2(s, H') \leq \frac{\sqrt{m}}{2M} \right\} ds$$

Thus, we can bound the probability that the commitment sampled according to Algorithm 4 belongs to the separating hyperplane H as follows:

$$\mathbb{P}(p \in H) = \mathbb{P}_{x \sim \mathcal{U}(\Xi)}(x \in H') \leq 2\frac{\int_{Q \cap H'} \frac{\sqrt{m}}{2M} dA}{\text{vol}_{m-1}(Q)}$$

$$= \frac{\sqrt{m}}{M} \frac{\int_{Q \cap H'} dA}{(2 + 1/M)^{m-1}}$$

$$\leq \frac{\sqrt{2m}}{M} \frac{(2 + 1/M)^{m-2}}{(2 + 1/M)^{m-1}} = \frac{\sqrt{m}}{\sqrt{2}M} \leq \delta.$$

Where 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 $\mathbb{P}(p \in H) \leq \delta$. Furthermore, we notice that for each linear space $\widetilde{H}$ with a dimension less than $m - 1$, there always exists a hyperplane H of dimension $m - 1$ such that $\widetilde{H} \subseteq H$ and thus:

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

In the following, we show that the point p sampled according to Algorithm 4 belongs to the polytope $\mathcal{P}$. To do that, we denote the h-vertex of the set $\mathcal{V}$ as v^h, so that $\mathcal{V} = \{v^1, \ldots, v^m\}$, with $V = |\mathcal{V}|$. Furthermore, given that $p^\diamond \in \text{int}(\mathcal{P})$, the $\|\cdot\|_2$-distance between $p^\diamond$ and any separating hyperplane H_{jk} that defines the boundary of $\mathcal{P}$ can be lower-bounded as follows:

$$d_2(p^\diamond, H_{jk}) = \left| \frac{\sum_{i=1}^m p_i^\diamond \left(u_f(a_i, a_j) - u_f(a_i, a_k) \right)}{\sqrt{\sum_{i=1}^m \left(u_f(a_i, a_j) - u_f(a_i, a_k) \right)^2}} \right|$$

$$= \left| \frac{\sum_{i=1}^m \sum_{h=1}^m \left(u_f(a_i, a_j) - u_f(a_i, a_k) \right) v_i^h}{m \sqrt{\sum_{l=1}^m \left(u_f(a_i, a_j) - u_f(a_i, a_k) \right)^2}} \right|$$

$$\geq \left(m^2 2^{9m^3 L + 2mL} \right)^{-1} =: r.$$

To prove the last inequality, we provide a lower bound for the numerator and an upper bound for the denominator of the fraction above. More specifically, the denominator can be upper bounded by m^2 since $|u_f(a_i, a_j) - u_f(a_i, a_k)| \leq 1$ for each $i \in [m]$. 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:

$$\left| \sum_{i=1}^m \sum_{h=1}^m \left(u_f(a_i, a_j) - u_f(a_i, a_k) \right) v_i^h \right| = \left| \sum_{i=1}^m \sum_{h=1}^m \frac{\beta_i \mu_i^h}{\gamma_i \nu_h} \right|$$

$$
= \left| \frac{\displaystyle\sum_{i=1}^{m}\sum_{h=1}^{m}\beta_i\mu_i^h\left(\prod_{i'\neq i}\gamma_{i'}\prod_{h'\neq h}\nu_{h'}\right)}{\displaystyle\prod_{i=1}^{m}\gamma_i\prod_{h=1}^{m}\nu_h} \right|
$$

$$
\geq \left(\prod_{i=1}^{m}\gamma_i\prod_{h=1}^{m}\nu_h\right)^{-1} \geq 2^{-(9m^3L+4mL)}.
$$

We observe that the first inequality holds because the numerator defining the above fraction can be lower bounded by one. On the other hand, the denominator can be upper bounded by noting that the bit complexity of γ_i is bounded by $4L$ for each $i \in [m]$, as ensured by Lemma D.1. Additionally, for every $h \in [m]$, the bit complexity of ν_h is bounded by $9m^2L$ thanks to Lemma D.2. With a similar argument, we can show that the 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:

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

$$
\leq \sqrt{\sum_{i\in[m-1]}\left(p_i^\diamond + \rho x_i - p_i^\diamond\right)^2 + \left(\sum_{i\in[m]}p_i^\diamond + \sum_{i\in[m-1]}\rho x_i - 1\right)^2}
$$

$$
\leq \sqrt{\sum_{i\in[m-1]}\left(\rho x_i\right)^2 + \left(\sum_{i\in[m-1]}\rho x_i\right)^2}
$$

$$
\leq \sqrt{(m-1)\rho^2 + (m-1)^2\rho^2} = \rho\sqrt{(m-1)m} < \rho m
$$

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

We also show that the bit complexity of p is bounded by $40m^3L + 2\log_2(1/\delta)$. 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:

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

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:

$$
B_D = \left\lceil \log_2\left(m\prod_{h=1}^{m}\nu_h D_\rho M\right)\right\rceil
$$

$$
= \left\lceil \log_2(m) + \sum_{h=1}^{m}\log_2(\nu_h) + \log_2(m2^{9m^3L+4mL}) + \log_2(M)\right\rceil
$$

$$
\leq 1 + \log_2(m) + \sum_{h=1}^{d}\log_2(2^{9m^2L}) + \log_2(m2^{9m^3L+4mL}) + \log_2(\sqrt{m}/\delta + 1)
$$

$$\leq 20m^3 L + \log_2(1/\delta).$$

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

D   ADDITIONAL TECHNICAL LEMMAS

Lemma D.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 $q \coloneqq \sum_{i=1}^m q_i$ 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 α_i / β_i, where both $\alpha_i, \beta_i \in \mathbb{Z}$ for each $i \in [m]$. Then the following holds:

$$q = \sum_{i=1}^m q_i = \sum_{i=1}^m \frac{\alpha_i}{\beta_i} = \frac{\sum_{i=1}^m \alpha_i \left(\prod_{i \neq j} \beta_j \right)}{\prod_{i \in [m]} \beta_i}.$$

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:

$$B_q \leq \log_2(m) + B_{\alpha_{i^*}} + \sum_{i \neq i^*} B_{\beta_i} + \sum_{i \in [m]} B_{\beta_i}$$

$$\leq \log_2(m) + B + 2 \sum_{i \neq i^*} B_{\beta_i}$$

$$\leq \log_2(m) + (2m - 1)B \leq 3mB.$$

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 D.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 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 H' 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 D.1, and by observing that the coefficients that defines the hyperplanes H' 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:

$$|\det(A')| \leq \prod_{i \in [m]} \sqrt{\sum_{j \in [m]} {a'_{ji}}^2} \leq \prod_{i \in [m]} \sqrt{\sum_{j \in [m]} (2^{4Lm})^2} = \prod_{i \in [m]} \sqrt{m(2^{4Lm})^2} = (2^{4Lm^2}) m^{\frac{m}{2}}.$$

Furthermore, observing that the same upper bound holds for $|\det(A^j)|$, the bit complexity of v is bounded by:

$$2 \left\lceil \log_2(2^{4Lm^2} m^{m/2}) \right\rceil \leq 2(\log_2(2^{4Lm^2} m^{m/2}) + 1) = 8Lm^2 + m \log_2(m) + 2 \leq 9Lm^2.$$

Finally, we notice that this upper bound holds when every component of the vertex has the same denominator $\det(A')$, concluding the proof. $\qquad \square$

E ON THE RUNNING TIME OF ALGORITHM 1

Theorem E.1. *With probability at least $1 - \zeta$, the running time of Algorithm 1 is polynomial when either the number of follower's actions n or that of leader's actions m is fixed.*

Proof. We observe that, by employing the same argument used to prove Theorem 4.3, before entering the while loop at Line 10, the set $\mathcal{C}$ is either empty or is such that if $a_k \in \mathcal{C}$, then with a probability of at least $1 - \zeta$, $\text{vol}(\mathcal{U}_k) > 0$, and $\mathcal{U}_k = \mathcal{P}_k$. Consequently, with this probability, the set $\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_{a_k}$ is always non-empty if Algorithm 1 has not terminated.

In the following, we show that for each set of actions $\mathcal{C}$ computed through the execution of Algorithm 1, we can always efficiently sample a point from the interior of $\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_{a_k}$ if either n or m is fixed. To achieve this, we define for each action $a_j \in \mathcal{C}$ the set $U(j) \subseteq A_f$ as $U(j) = \{a_k \in A_f \mid a_k \neq a_j \wedge H_{jk} \cap \mathcal{P}_j \text{ is a facet of } \mathcal{P}_j\} = \{a_k \in A_f \mid a_k \neq a_j \wedge \text{vol}_{m-2}(H_{jk} \cap \mathcal{P}_j) > 0\}$. Furthermore, we let $P_j := \bigcup_{a_k \in U(j)} (j, k)$ and $P := \bigtimes_{a_j \in \mathcal{C}} P_j$. Observe that an element $\rho \in P$ indicates a pair (j, k) for every $a_j \in \mathcal{C}$. We denote with $\rho(j)$ the pair (j, k) relative to action a_j. As a result, the following holds:

$$\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{P}_k = \Delta_m \setminus \bigcup_{a_j \in \mathcal{C}} \mathcal{U}_j$$

$$= \Delta_m \setminus \bigcup_{a_j \in \mathcal{C}} \left(\mathcal{U}'_j \cap \Delta_m \right)$$

$$= \Delta_m \setminus \left(\left(\bigcup_{a_j \in \mathcal{C}} \mathcal{U}'_j \right) \cap \Delta_m \right)$$

$$= \Delta_m \cap \left(\bigcup_{a_j \in \mathcal{C}} \mathcal{U}'_j \right)^C$$

$$= \Delta_m \cap \Big(\bigcap_{a_j \in \mathcal{C}} \mathcal{U}_j'^C \Big)$$

$$= \Delta_m \cap \Big(\bigcap_{a_j \in \mathcal{C}} \Big(\bigcap_{a_k \in U(j)} \mathcal{H}_{jk} \Big)^C \Big)$$

$$= \Delta_m \cap \Big(\bigcap_{a_j \in \mathcal{C}} \Big(\bigcup_{a_k \in U(j)} \mathcal{H}_{jk}^C \Big) \Big)$$

$$= \Delta_m \cap \bigcup_{\rho \in P} \Big(\bigcap_{a_j \in \mathcal{C}} \mathcal{H}_{\rho(j)}^C \Big)$$

$$= \bigcup_{\rho \in P} \Big(\bigcap_{a_j \in \mathcal{C}} \mathcal{H}_{\rho(j)}^C \cap \Delta_m \Big)$$

$$= \bigcup_{\rho \in P} \widetilde{\mathcal{U}}_\rho,$$

where the equalities above follow by iteratively employing De Morgan's laws. Thus, the set $\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_k$ is non-empty with probability at least $1 - \zeta$, as observed before, and it can be decomposed into $|P|$ convex sets $\widetilde{\mathcal{U}}_\rho$ with $\rho \in P$. As a result, we can employ Algorithm 4, since $\widetilde{\mathcal{U}}_\rho$ is convex. Consequently, we can select the $\rho \in P$ such that $\widetilde{\mathcal{U}}_\rho$ has non-empty interior. Since the set P has a size of at most $\mathcal{O}(n^n)$, we can enumerate all the elements of P in polynomial time, when the number of follower's actions n is fixed.

Finally, for every region $\widetilde{\mathcal{U}}_\rho$ with non-empty interior, for $\rho \in P$, there exists a subset of its vertices V with dimension m, ensuring $\text{co}(V) \subseteq \widetilde{\mathcal{U}}_\rho$ and $\text{vol}(co(V)) > 0$. Consequently, by enumerating all possible subsets of the vertices defining the regions $\widetilde{\mathcal{U}}_\rho$ with m elements, and observing that the number of such vertices is at most $\binom{m+n^2}{m} \leq (m+n^2)^m$, we are required to check a maximum of $\binom{(m+n^2)^m}{m}$ regions defined by a set of these m vertices. This results in a complexity of $O(n^{2m^2})$ when m is constant. This enumeration allows us to compute a set V containing m linearly independent vertices of a polytope $\widetilde{\mathcal{U}}_\rho$ for some $\rho \in P$. We observe that Algorithm 4 actually needs just m linearly independent vertices of the polytope $\mathcal{P}$ (see line 2) and no other parameter. Thus, we can execute `Sample-Int`$(\widetilde{\mathcal{U}}_\rho)$ by assigning $\mathcal{V} \leftarrow V$ at line 2 Algorithm 4, where V is the set of m linearly independent vertices found above.

As a result, since sampling from the set $\Delta_m \setminus \bigcup_{a_k \in \mathcal{C}} \mathcal{U}_{a_k}$ can be efficiently done when either n or m is constant, and noting that the running time of all other steps in Algorithm 1 can be performed in polynomial time, we deduce that the overall running time of Algorithm 1 is polynomial if either the number of follower's or leader's actions is a fixed parameter. □

F EXTENSION TO THE CASE WITH EQUIVALENT FOLLOWER'S ACTIONS

In this section, we show how to extend our results to the general case in which there could equivalent follower's actions, where two actions $a_j, a_k \in A_f$ are *equivalent* if $u_f(a_i, a_j) = u_f(a_i, a_k)$ for all $a_i \in A_\ell$.

Our algorithm can be easily extended to handle such a case. To do this, we introduce the following additional notation. First, we introduce the notion of *leader separating hyperplane*. More formally, given a pair of follower's actions $a_j, a_k \in A_f$ such that $a_j \neq a_k$, we let $\mathcal{H}_{jk}^\ell \subseteq \mathbb{R}^m$ be the halfspace in which a_j is (weakly) better than a_k in terms of leader's utility, where:

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

Furthermore, we let $H_{jk}^\ell := \partial \mathcal{H}_{jk}^\ell$ the hyperplane defining the boundary of the halfspace $\mathcal{H}_{jk}^\ell$, which we call the leader separating hyperplane between a_j and a_k. In addition, given a follower's action $a_j \in A_f$, with an abuse of notation we define $A_f(a_j)$ as the set of follower's actions $a_k \in A_f : a_j \neq a_k$ that are equivalent to a_j. Formally:

$$A_f(a_j) := \{a_k \in A_f \mid a_k \neq a_j \wedge u_f(a_i, a_j) = u_f(a_i, a_k) \ \forall a_i \in \mathcal{A}_l\}.$$

Then, we can re-define the best-response region of action $a_j \in A_f$ as:

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

Intuitively, the region $\widetilde{\mathcal{P}}_j$ corresponds to the region in which a_j is a follower's best response when also accounting for tie-breaking, which is done in favor of the leader. Consequently, we can apply Algorithm 1 and achieve equivalent results. This is because the regions $\widetilde{\mathcal{P}}_j$ are still polytopes, with the only difference being that the number of vertices defining them is bounded by $\binom{m+2n-1}{m}$, as the number of hyperplanes that define their boundary also takes into account leader separating hyperplanes. Furthermore, by letting L be the bit-complexity of leader's payoffs, it is possible to restate Theorem 4.3 as:

Theorem F.1. *Given any $\zeta \in (0,1)$, with probability at least $1 - \zeta$, Algorithm 1 terminates with $p^\star$ being an optimal strategy to commit to, by using a number of samples of the order of $\widetilde{\mathcal{O}}\left(n^2 \left(m^7 L \log(1/\zeta) + \binom{m+2n}{m}\right)\right)$,*

Theorem F.1 shows that we can achieve similar results to those obtained in Theorem 4.3, even in cases where there equivalent follower's actions are allowed, by using Algorithm 1. Specifically, when the number of leader's actions is fixed, the number of samples required to compute an optimal commitment is of the same order as when there are no equivalent follower's actions. In contrast, when the number of follower's actions is fixed, the required number of samples is $\widetilde{\mathcal{O}}(m^{2n})$, differently from the case with no coinciding follower's actions, where it is equal to $\widetilde{\mathcal{O}}(m^n)$.