Learning and Approximating the Optimal Strategy to Commit To[*]

Joshua Letchford, Vincent Conitzer, and Kamesh Munagala

Department of Computer Science, Duke University, Durham, NC, USA
{jcl,conitzer,kamesh}@cs.duke.edu

Abstract. Computing optimal Stackelberg strategies in general two-player Bayesian games (not to be confused with Stackelberg strategies in routing games) is a topic that has recently been gaining attention, due to their application in various security and law enforcement scenarios. Earlier results consider the computation of optimal Stackelberg strategies, given that all the payoffs and the prior distribution over types are known. We extend these results in two different ways. First, we consider *learning* optimal Stackelberg strategies. Our results here are mostly positive. Second, we consider computing *approximately* optimal Stackelberg strategies. Our results here are mostly negative.

1 Introduction

Game theory defines solution concepts for strategic situations, in which multiple self-interested agents interact in the same environment. Perhaps the best-known solution concept is that of *Nash equilibrium* [11]. A Nash equilibrium prescribes a strategy for every player, in such a way that no individual player has an incentive to change her strategy. If strategies are allowed to be mixed—a mixed strategy is a probability distribution over pure strategies—then it is known that every finite game has at least one Nash equilibrium. Some games have more than one equilibrium, leading to the *equilibrium selection problem*.

Perhaps the most basic representation of a game is the *normal form*. In the normal-form representation, every player's pure strategies are explicitly listed, and for every combination of pure strategies, every player's utility is explicitly listed.

The problem of *computing* Nash equilibria of a normal-form game has received a large amount of attention in recent years. Finding a Nash equilibrium is PPAD-complete [6, 1]. Finding an optimal equilibrium (for just about any reasonable definition of "optimal"—for instance, maximizing the sum of the players' utilities) is NP-hard [7, 3]; moreover, it is not even possible to find an equilibrium that is approximately optimal in polynomial time, unless P=NP [3]. This holds even for two-player games.

However, Nash equilibrium is not always the right solution concept. In some settings, one player can credibly commit to a strategy, and communicate this to the other player, before the other player can make a decision. To see how this can affect the

[*] Some of these results were briefly presented as part of a talk at the 2009 Bellairs Workshop on Algorithmic Game Theory. This work is funded by: Alfred P. Sloan Research Fellowships, NSF Grant IIS-0812113, NSF Career Award 0745761 and Grant CNS-0540347.

outcome of a game, consider the following simple normal-form game (which has previously been used as an example for this, *e.g.*, [2]):

	L	R
U	(2,1)	(4,0)
D	(1,0)	(3,1)

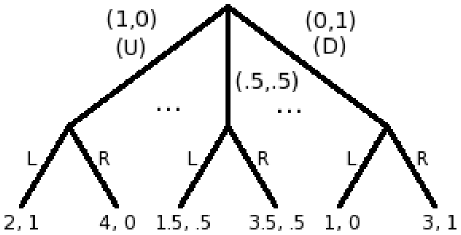

Fig. 1. A sample game and its extensive form representation

For the case where the players move simultaneously (no ability to commit), the unique Nash equilibrium is (U, L): U strictly dominates D, so that the game is solvable by iterated strict dominance. So, player 1 (the row player) receives utility 2. However, now suppose that player 1 has the ability to commit. Then, she is better off committing to play D, which will incentivize player 2 to play R, resulting in a utility of 3 for player 1. The situation gets even better for player 1 if she can commit to a mixed strategy: in this case, she can commit to the mixed strategy $(.5 - \epsilon, .5 + \epsilon)$, which still incentivizes player 2 to play R, but now player 1 receives an expected utility of $3.5 - \epsilon$. To ensure the existence of optimal strategies, we assume (as is commonly done [2, 12]) that player 2 breaks ties in player 1's favor, so that the optimal strategy for player 1 to commit to is $(.5, .5)$, resulting in a utility of 3.5. (Note that there is never a reason for player 2 to randomize, since he effectively faces a single-agent decision problem.) An optimal strategy to commit to is usually called a *Stackelberg* strategy, after von Stackelberg, who showed that in Cournot's duopoly model [4], a firm that can commit to a production quantity has a strategic advantage [15]. Throughout this paper, a Stackelberg strategy is an optimal *mixed* strategy to commit to; we will only consider two-player games. In this context, the Stackelberg leader's expected utility is always at least the expected utility that she would receive in any Nash (or even correlated) equilibrium of the simultaneous-move game [16]. In contrast, committing to a pure strategy is not always beneficial; for example, consider matching pennies.

One may argue that the normal form is not the correct representation for this game. In game theory, the time structure of games is usually represented by the *extensive form*. Indeed, the above game can be represented as the extensive-form game in Figure 1. While this is a conceptually useful representation, from a computational perspective it is not helpful: player 1 has an infinite number of strategies, hence (the naïve representation of) the tree has infinite size. It should be emphasized that committing to a mixed strategy is *not* the same as randomizing over which pure strategy to commit to; in fact, there is no reason to randomize over which strategy to commit to. Thus, from a computational viewpoint, it makes more sense to operate directly on the normal form.

The problem of computing Stackelberg strategies in general normal-form (or, more generally, Bayesian) games has only recently started to receive attention. A 2006 EC paper by Conitzer and Sandholm [2] layed out the basic complexity results for this setting: Stackelberg strategies can be computed in polynomial time for two-player general-sum normal-form games using linear programming (in contrast to the problem of finding a Nash equilibrium), but computing Stackelberg strategies is NP-hard for two-player Bayesian games or three-player normal-form games. Undeterred by the NP-hardness result, Paruchuri *et al.* [12] developed a mixed-integer program for finding an (optimal) Stackelberg strategy in the two-player Bayesian case (the setting that we study in this paper). They show that using this formulation is much faster than converting the game to normal form (leading to an exponential increase in size) and then using the linear programming approach. Moreover, this algorithm forms the basis for their deployed ARMOR system, which is used at the Los Angeles International Airport to randomly place checkpoints on roads entering the airport, as well as to decide on canine patrol routes [9, 13]. The use of commitment in similar games dates back much further, including, for example, applications to inspection games [10]. The formal properties of various types of commitment are also studied in [8].

It should be noted that Stackelberg strategies are a generalization of minimax strategies in two-player zero-sum games. Because computing minimax strategies is equivalent to linear programming [5], this also implies that a linear programming solution for computing Stackelberg strategies is the best that we can hope for. Of course, Nash equilibrium is an alternative generalization of minimax strategies. Stackelberg strategies have the significant advantage that they avoid the equilibrium selection problem: there is an optimal value of the game for the leader (player 1), which in general corresponds to a single optimal strategy (though not in degenerate cases). The notion of "Stackelberg strategies" has appeared in other contexts in the algorithmic game theory literature, specifically, in the context of routing games, where a single benevolent party controls part of the flow, and commits to routing this flow in a manner that minimizes total latency [14]. While interesting, that paper does not seem that closely related to our work, because in our context, the leader is a selfish player in an arbitrary game.

The rest of this paper is layed out as follows. In Section 2, we formally review the necessary concepts, introduce our notation, and discuss existing results that are relevant. In Section 3—the first half of our contribution—we prove several results about *learning* Stackelberg strategies, in contexts where the follower payoffs and/or the distribution over types is not known initially. In Section 4—the second half of our contribution—we consider purely computational problems and give (in)approximability results.

2 Preliminaries

In this section, we review notation and existing results.

2.1 Notation and definitions

We will refer to player 1 as the *leader* and to player 2 as the *follower*. Let A_l be the set of leader actions in the game ($|A_l| = d$), and let A_f be the set of follower actions ($|A_f| = k$). The leader's utility is given by a function $u_l : A_l \times A_f \to \mathbb{R}$. When we are studying approximability, we (wlog) require all the leader utilities to be nonnegative (to make multiplicative approximation meaningful). In a Bayesian game, the follower has a set of *types* Θ ($|\Theta| = \tau$), which, together with the actions taken, determine his utility, according to a function $u_f : \Theta \times A_l \times A_f \to \mathbb{R}$. For simplicity, we will not consider situations where the leader's utility also depends on the follower's type; this restriction strengthens our hardness results. We will refer to these as *Bayesian* games; a *normal-form* game is the special case where there is only a single type.

σ denotes a mixed strategy for the leader, and $\sigma(a_l)$ the probability that σ places on action a_l. Let $\mathrm{BR}(\theta, \sigma) \in A_f$ denote the action that the follower plays (that is, his best response, with ties broken in favor of the leader) when his type is θ and the leader has committed to playing σ. We note that

$$\mathrm{BR}(\theta, \sigma) \in \arg\max_{a_f \in A_f} \sum_{a_l \in A_l} \sigma(a_l) u_f(\theta, a_l, a_f)$$

The BR function also captures the fact that the follower breaks ties in the leader's favor. Given the follower type θ, the leader's expected utility is

$$\sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, \mathrm{BR}(\theta, \sigma))$$

Given a prior probability distribution $P : \Theta \rightarrow [0, 1]$ over follower types, the leader's expected utility for committing to σ is

$$\sum_{\theta \in \Theta} P(\theta) \sum_{a_l \in A_l} \sigma(a_l) u_l(a_l, \mathrm{BR}(\theta, \sigma))$$

When we take a worst-case perspective, we will be interested in a setting with types but without a prior distribution over them (also known as a *pre-Bayesian* game).

2.2 Known results and techniques

In this subsection we review the most relevant prior work. For a normal-form game, the optimal mixed leader strategy can be computed in polynomial time, as follows:[1] for every follower action a_f, the following linear program (whose variables are the $\sigma(a_l)$) can be used to determine the best leader strategy that makes the follower play a_f:

$$\boxed{\begin{array}{l} \textbf{maximize } \sum_{a_l} \sigma(a_l) u_l(a_l, a_f) \\ \textbf{subject to} \\ (\forall a'_f)\ \sum_{a_l} \sigma(a_l) u_f(a_l, a_f) \geq \sum_{a_l} \sigma(a_l) u_f(a_l, a'_f) \\ \sum_{a_l} \sigma(a_l) = 1 \\ (\forall a_l)\ \sigma(a_l) \geq 0 \end{array}}$$

Some of these linear programs may be infeasible (it is impossible to make a follower play a strictly dominated strategy), but some will be feasible; the solution of the one with the highest objective value gives the optimal mixed strategy for the leader.

For Bayesian games (with a prior), the problem of computing the optimal mixed leader strategy is known to be NP-hard [2]. However, this strategy can be found using a mixed integer program [12].

[1] This algorithm was presented in [2]. Some of the analysis in [16] is based on similar insights.

2.3 Visualization

In this subsection, we show how the problems we discussed above can be visualized. Let us consider the normal-form case. The space of possible strategies for the leader defines a unit simplex in $d - 1$ dimensions, where d is the number of leader actions. For each strategy of the leader, the follower has a best response. The space of leader strategies for which the follower's best response is a_f defines a (possibly empty) polyhedron. Therefore, the d-simplex splits into at most k (number of follower actions) polyhedral regions, based on the follower utility function. Each of these regions corresponds to the feasible region of one of the linear programs, and the objective of that linear program can be represented as an arrow in the region.

Let us consider the following small example and its visualization.

	L	C	R
U	(0,1)	(1,0)	(0,0)
M	(4,0)	(0,1)	(0,0)
D	(0,0)	(1,0)	(1,1)

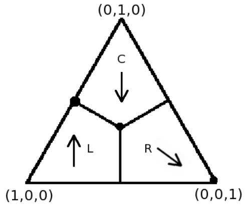

Fig. 2. A small game and its visualization

Each dot in Figure 2 represents the optimal point (leader mixed strategy) within each region (which lie on *separating hyperplanes* or on the boundary); the largest dot (.5,.5,0) shows the optimal point overall.

The Bayesian case can be visualized in (at least) two different ways. A simple way is to have a separate unit simplex for every type; this does not require a prior distribution over types (that is, it works for pre-Bayesian games). If there is a prior distribution over types, another way is to have a region for each element of the set of all pure strategies for the follower, so that $(a_f^{\theta^1}, \ldots, a_f^{\theta^\tau})$ corresponds to the region where type θ^1's best response is $a_f^{\theta^1}$, type θ^2's best response is $a_f^{\theta^2}$, *etc.* The arrows in this region represent the objective, which depends on the prior. This representation does not work for pre-Bayesian games where we take a worst-case perspective, because the optimal point may be in the interior of a region.

3 Learning Stackelberg strategies

If a game is repeated over time, this opens up the possibility for the leader to learn something about the follower's utilities or the distribution over types. To avoid the possibility that the follower tries to mislead the leader over time, we imagine that a new follower agent is drawn in every round. Alternatively, the follower can be assumed to behave myopically. In a round, the leader commits to a mixed strategy, and subsequently observes the follower's response. The leader's goal is to learn enough to determine the optimal Stackelberg strategy, in as few rounds (*samples*) as possible.

Due to space constraint, we focus on the case with a single type: that is, in each round, the follower has the same payoff matrix, given by $u_f(a_l, a_f)$, initially unknown to the leader. In each round, the leader commits to a mixed strategy σ and learns the follower's response. We say that the leader *queries* or *samples* the point σ on the probability simplex. The goal is to minimize the number of samples necessary to find the optimal (Stackelberg) mixed strategy for the leader. In Appendices B and C we consider two other cases with more than one type, one where the leader needs to learn the follower payoff function, and one where this function is known, but the leader must discover the distribution over types. We make the following assumptions:

- The follower utilities are non-degenerate; no separating hyperplanes coincide.
- We will only consider regions whose volume is at least some fraction $\epsilon > 0$ of the total volume, and try to find the optimal solution among points in these regions. (It can be argued that solutions in smaller regions are too unstable. Alternatively, we can simply assume that every nonempty region has at least this volume.)
- We assume that the optimal solution can be specified exactly using a limited amount of precision quantified by L. This allows us to bound the number of iterations of binary search needed to calculate these hyperplanes exactly, to a linear multiple of L.

Our approach will be to learn all the regions (whose volume is at least ϵ of the total)—that is, find all hyperplanes separating these regions. Once we know these, the optimal strategy can be computed using the linear programming approach above.

A high-level outline of our algorithm SU is as follows. For each follower action $a_f \in A_f$, the algorithm maintains an overestimate P_{a_f} of the region where a_f is a best response. It then refines these overestimates via sampling, until they are disjoint.

```
SU

1. For each $a_f \in A_f$, find a point (leader strategy) $q_{a_f}$ in the $d$-simplex to which $a_f$
   is a best response (provided the corresponding region is sufficiently large).
2. Initially, each $P_{a_f}$ is the entire $d$-simplex.
3. Repeat the following until all $P_{a_f}$ are disjoint:
   (a) Find a point $p^*$ in the intersection of some $P_{a'_f}$ and $P_{a''_f}$.
   (b) Sample to obtain the optimal follower strategy at $p^*$; call it $a^*_f$.
   (c) Draw a line segment between $p^*$ and some $q_{a_f}$ for $a_f \neq a^*_f, a_f \in \{a'_f, a''_f\}$;
       perform binary search on this line to find a single point on a hyperplane that
       we have not yet discovered.
   (d) Find a set of $d$ linearly independent points on the hyperplane, and hence reconstruct it.
   (e) Update the $P_{a_f}$ to take this new hyperplane into account.
```

We now describe the steps of SU in detail.

Step (1). Finding a point in each region (with at least ϵ of the volume) can be achieved via random sampling, via the following lemma.

Lemma 1. *It takes $O(Fk \log k)$ samples to w.h.p. (with high probability) find a single point in each sufficiently large region, where $F = 1/\epsilon$.*

Proof. The probability that a randomly chosen point corresponds to follower action a_f is at least ϵ. Therefore, for any constant integer $c \geq 1$, after $((c+1)F \log k)$ samples, the probability that follower action a_f is not hit is at most $(\frac{1}{k})^{c+1}$. By a union bound, the probability that at least one action is not hit is at most $(\frac{1}{k})^{c}$.

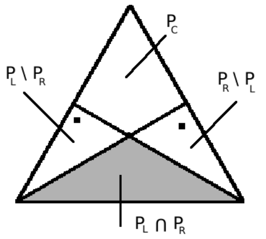

Fig. 3. Finding a hyperplane.

Step (3 a–c). Consider two overestimates $P_{a'_f}$ and $P_{a''_f}$ that have nonzero overlap volume. By Step (1), we may assume that we have sampled a point $q_{a'_f}$ that led to a response of a'_f (that is, $q_{a'_f}$ is in the region corresponding to a'_f), and a point $q_{a''_f}$ that led to a response of a''_f. Both of these overestimates are characterized by sets H' and H'' of hyperplanes that we have previously discovered. We need to discover a new hyperplane. It will not suffice to do binary search on the line segment between the two starting points, as illustrated by Figure 3, which illustrates a situation where we have discovered two of the hyperplanes of Figure 2. If we do binary search on the line segment between the two indicated points, we cannot discover the missing hyperplane, because the top region "gets in the way" (another action, namely C, will start being the best response). However, if we sample from the shaded set $P_L \cap P_R$, the result will be different from one of the two points; then, by performing binary search on the line segment between this point and the new point, we will find a point on a new hyperplane. The following algorithm formalizes this idea. In it, we do not assume that the two overestimates overlap.

```
FIND POINT

1. Solve a linear program to find an interior point $p^*$ of $P_{a'_f} \cap P_{a''_f}$ given the constraints $H' \cup H''$. (If this is not feasible, return failure.)
2. Sample this point and let the follower strategy returned be $a^*_f$.
   (a) If $a^*_f = a'_f$, search the line segment between $p^*$ and $q_{a''_f}$ for a point on a hyperplane that has the region corresponding to $a''_f$ adjacent on one side, via binary search.
   (b) Otherwise, search the line segment between $p^*$ and $q_{a'_f}$ for a point on a hyperplane that has the region corresponding to $a'_f$ adjacent on one side, via binary search.
```

Lemma 2. *Given overestimates $P_{a'_f}$ and $P_{a''_f}$ on the regions corresponding to a'_f and a''_f, and points $q_{a'_f}$ and $q_{a''_f}$ in these respective regions,* FIND POINT *will either give a* *point on a new hyperplane for one of the regions $P_{a'_f}$ or $P_{a''_f}$, or will return that $P_{a'_f}$ and $P_{a''_f}$ already have zero intersection volume. This requires $O(L)$ samples.*

The detailed proof is in Appendix A.

Step (3d). In this step, the input is a point p on the hyperplane that we need to reconstruct, and the two follower actions a'_f and a''_f that correspond to the regions separated by this hyperplane. The following DETERMINE HYPERPLANE finds the hyperplane.

```
DETERMINE HYPERPLANE

1. Sample the vertices of a regular $d$-simplex with sides of length $\epsilon' \ll \epsilon$, centered at $p$. (Draw this simplex uniformly at random among such simplices.)
2. Organize the vertices of this simplex into two sets, $V'$ and $V''$ according to the region they fall in. (Both of these sets will be nonempty.)
3. Choose $d$ distinct pairs of points where one of the points is in $V'$ and the other is in $V''$
4. Binary-search the $d$ line segments formed by these pairs, to find the points where these line segments intersect the hyperplane.
```

Lemma 3. DETERMINE HYPERPLANE *will give d linearly independent points on the hyperplane using $O(dL)$ samples.*

Proof. First, consider the $d + 1$ vertices of the d-simplex centered at p. Since ϵ' is sufficiently small, all of the points fall into one of the two regions (and since the simplex is chosen at random, there is zero probability of one of the vertices being exactly on the hyperplane). Since the hyperplane goes through p, at least one of the vertices of the simplex will fall into each region. As a result, there are at least d line segments between vertices of the simplex where the two vertices of the segment produce different follower actions. Finally, the points where the hyperplane intersects with these line segments must be linearly independent; otherwise, the simplex would not be full-dimensional. Furthermore, the number of samples needed to find the hyperplane-intersecting point on a line segment via binary search is linear in L. This completes the proof.

With these tools, we can give our main result for this problem:

Theorem 1. *To find, w.h.p., all the hyperplanes that separate regions, SU requires $O(Fk \log k + dk^2 L))$ samples, where $F = 1/\epsilon$, ϵ is the smallest volume of regions that we consider, L is the precision, and $k = |A_f|$. Computationally, this requires the solution of $O(k^2)$ linear programs.*

Details of the proof are in Appendix A. Once we have generated all the hyperplanes that separate regions, we can use the known linear programming approach described in Subsection 2.2 to find the optimal mixed strategy to commit to.

4 Computing Stackelberg strategies

In this section, we consider how different modeling assumptions affect the computational tractability and approximability of the Stackelberg problem with multiple follower types. Unlike the previous section, this section does not consider learning problems at all: it focuses strictly on the computational aspects of the optimization. Because of this, we only consider a single-round setting in this section.

The following aspects of the model will remain the same throughout this section.

- We consider two-player, general-sum games that have more than one follower type.
- The leader's utility does not depend *directly* on the follower's type (but it does depend on the follower's action, which can be affected by the follower's type).
- The follower's utility function $u_f(\theta, a_l, a_f)$ is common knowledge.

We consider two modeling decisions. The first decision concerns whether the type space is discrete or continuous. For the discrete case, we assume that we have a finite number of types, which are explicitly listed. For the continuous case, we assume that the space of possible types is defined by a lower bound and an upper bound for the follower's utility for each action profile (a_l, a_f); every follower payoff matrix that is consistent with these bounds corresponds to some type.

The second modeling decision is whether the follower type is chosen according to a Bayesian model or an adversarial (worst-case) model. Note that the "adversary" is *not* one of the players of the game, in particular, the adversary and the follower are different.

4.1 Computing Bayesian optimal strategies with finitely many types

In this subsection we study how to compute the optimal mixed strategy when the follower's type is drawn from a known distribution over finitely many types. We refer to this problem as *Bayesian optimization for finite types (BOFT)*. BOFT is defined as:

- We have a set Θ of possible follower types, $|\Theta| = \tau$.
- The follower's utility function $u_f(\theta, a_l, a_f)$ is common knowledge.
- Both the follower's utility function $u_f(\theta, a_l, a_f)$ and the leader's utility function $u_f(\theta, a_l, a_f)$ are normalized to lie in [0,1] for all inputs.
- The prior over follower types $P(\theta)$ is common knowledge.
- An optimal leader strategy is one that maximizes the leader's expected utility.

This problem was first studied in [2], where it was shown to be NP-hard. It also forms the basis for much of the applied work on computing Stackelberg strategies [9]. However, to the best of our knowledge, the approximability of this problem has not yet been studied. We settle the approximability precisely in this subsection.

Theorem 2. *For all constant $\epsilon > 0$, no polynomial-time factor-$\tau^{1-\epsilon}$ approximation exists for BOFT unless NP = P, even if there are only two follower actions.*

This hardness of approximation can be shown by a reduction from MAX-INDEPENDENT-SET. In this reduction, vertices correspond to types, and the leader cannot incentivize two adjacent types to both play a desirable action. The full reduction appears in Appendix D.

Theorem 3. *There is a polynomial-time factor-τ approximation algorithm for BOFT.*

A simple algorithm that achieves this is the following: choose a type uniformly at random, and solve for the optimal mixed strategy to commit to for this specific type (using the linear programming approach). With probability $1/\tau$, we choose the type that is actually realized, in which case we perform at least as well as the optimal overall strategy. Hence, this guarantees at least a τ approximation. Details and derandomization appear in Appendix D.

4.2 Computing worst-case optimal strategies with finitely many types

A prior distribution over follower types is not always readily available. In that case, we may wish to optimize for the worst-case type (equivalently, the worst-case distribution over types). We note that the worst-case type depends on the mixed strategy that we choose, so that this is not the same problem as optimizing against a single type. We refer to this problem as *worst-case optimization for finite types (WOFT)*:

- We have a set Θ of possible follower types, $|\Theta| = \tau$.
- The follower's utility function $u_f(\theta, a_l, a_f)$ is common knowledge.
- An optimal leader strategy is one that maximizes the worst-case expected utility for the leader, where the worst case is taken over follower types (but we are taking the expectation over the mixed strategy). That is, an adversary (not equal to the follower) chooses the follower type after the leader mixed strategy is chosen, but before the pure-strategy realization.

It turns out that WOFT is even less approximable than BOFT.

Theorem 4. *WOFT is completely inapproximable in polynomial time, unless P=NP (that is, it is hard to distinguish between instances where the leader can get at least* 1 *in the worst case, and instances where the leader can only get* 0*)—even if there are only four follower actions.*

This can be shown by a reduction from 3-SAT. In the resulting game, the leader can obtain an expected utility of 1 against every type if the 3-SAT instance is satisfiable, and otherwise will obtain utility 0 against some type. The full reduction appears in Appendix D.

4.3 Optimizing for the worst type with ranges

So far, we have assumed that the space of possible types is represented by explicitly listing the (finitely many) types and the corresponding utilities. However, this representation of the uncertainty that the leader has over the follower's preferences is not always convenient. For example, the leader may have a rough idea of every follower payoff, which could be represented by a range in which that payoff must lie. This corresponds to a continuous type space for the follower: every setting of all the follower payoffs within the ranges corresponds to a type.

In this subsection, we study the problem of maximizing the leader's worst-case utility over all types (instantiations of the follower payoffs within the ranges). Later in the subsection, we also consider a generalization where the follower payoffs in different entries can be linked to each other.

For example, consider the following game with ranges:

	L	R
U	0, [1,2]	1, 0
D	1,0	0, [1,2]

The leader is unsure about the follower's utility for (U, L) and (D, R), each of which is known to lie somewhere in the range $[1, 2]$ (they can vary independently). The follower knows his utilities. If the leader places less than $1/3$ probability on U, then the follower is guaranteed to play R; this results in a utility of at most $1/3$ for the leader. If the leader places more than $2/3$ probability on U, then the follower is guaranteed to play L; this results in a utility of at most $1/3$ for the leader. If the leader places probability between $1/3$ and $2/3$ on U, then the follower may end up playing either L or R; by placing probability $1/2$ on U, the leader obtains an expected utility of $1/2$, which is optimal.

We refer to this problem as *worst-case optimization for range types (WORT)*:

- For every (a_l, a_f), the leader has a range in which the follower utility might lie, $u_f(a_l, a_f) \in [u^l_f(a_l, a_f), u^h_f(a_l, a_f)]$. The leader knows her own utilities $u_l(a_l, a_f)$.
- An optimal leader strategy is one that maximizes the worst-case expected utility for the leader, where the worst-case values of

Theorem 5. *WORT is NP-hard.*

This follows from a reduction from 3-COVER, which is presented in Appendix D. It is an open question whether WORT can be efficiently approximated. In Appendix E, we define a generalization of WORT, which we prove is inapproximable unless $P = NP$. This generalization allows the follower's payoffs to be linked across entries.

5 Conclusion

Computing optimal Stackelberg strategies in general two-player Bayesian games is a topic that has been gaining attention in recent years, due to their application in both security and law enforcement. Earlier results consider the computation of optimal Stackelberg strategies, given that all the payoffs and the prior distribution over types are known. We extended these results in two ways.

First, we considered *learning* optimal Stackelberg strategies. We first considered the normal-form case where the follower payoffs are not known and showed how we can efficiently learn enough about the payoffs to determine the optimal strategy. We then extended this to Bayesian games. We also considered the case where the payoffs are known, but the distribution over types is not. We showed how we can efficiently learn enough about the distribution to determine the optimal strategy. It must be admitted that it is debatable whether this framework for learning is practical for current real-world security applications, since the costs incurred during the learning phase may be too high; however, these costs may be more manageable in electronic commerce applications.

Second, we considered computing *approximately* optimal Stackelberg strategies. Our results here were mostly negative: we showed that the best possible approximation ratio that can be obtained in polynomial time for the standard Bayesian problem is τ, the number of types, unless NP = P. Optimizing for the worst type is completely inapproximable in polynomial time, in the sense that we cannot distinguish instances where we can guarantee utility 1 from instances where it is impossible to guarantee positive utility, unless P=NP. We also studied a different representation of uncertainty about the follower's payoffs that relies on ranges, and showed that optimizing for the worst case is NP-hard in the basic setting, and completely inapproximable in a generalized setting where the payoffs are linked. These negative results provide some justification for the use of worst-case exponential-time algorithms in this context, such as those that use mixed integer programming.

Two immediate directions for future research are: (1) investigating the approximability of the basic ranges problem, and (2) considering the ranges problem in the

Bayesian case (rather than the worst case). There are many other directions for future research, for example, studying the number of samples required to learn *approximately* optimal strategies, investigating the case where there are more than two players, and/or computing optimal Stackelberg strategies when the normal form has exponential size, but the game is concisely represented.

References

1. Xi Chen and Xiaotie Deng. Settling the complexity of two-player Nash equilibrium. In *FOCS*, pages 261–272, 2006.
2. Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the ACM Conference of EC*, pages 82–90, Ann Arbor, MI, USA, 2006.
3. Vincent Conitzer and Tuomas Sandholm. New complexity results about Nash equilibria. *Games and Economic Behavior*, 63(2):621–641, 2008.
4. Antoine Augustin Cournot. *Recherches sur les principes mathématiques de la théorie des richesses (Researches into the Mathematical Principles of the Theory of Wealth),* 1838.
5. George Dantzig. A proof of the equivalence of the programming problem and the game problem. In Tjalling Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 330–335. John Wiley & Sons, 1951.
6. Constantinos Daskalakis, Paul Goldberg, and Christos H. Papadimitriou. The complexity of computing a Nash equilibrium. In *STOC*, pages 71–78, 2006.
7. Itzhak Gilboa and Eitan Zemel. Nash and correlated equilibria: Some complexity considerations. *Games and Economic Behavior*, 1:80–93, 1989.
8. Paul Harrenstein, Felix Brandt, and Felix Fischer. Commitment and extortion. In *Proceedings of AAMAS*, Honolulu, HI, USA, 2007.
9. Manish Jain, James Pita, Milind Tambe, Fernando Ordóñez, Praveen Paruchuri, and Sarit Kraus. Bayesian Stackelberg games and their application for security at Los Angeles international airport. *SIGecom Exch.*, 7(2):1–3, 2008.
10. Michael Maschler. A price leadership method for solving the inspector's non-constant-sum game. *Naval Research Logistics Quarterly*, 13(1):11–33, 1966.
11. John Nash. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36:48–49, 1950.
12. Praveen Paruchuri, Jonathan P. Pearce, Janusz Marecki, Milind Tambe, Fernando Ordóñez, and Sarit Kraus. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of AAMAS*, pages 895–902, Estoril, Portugal, 2008.
13. James Pita, Manish Jain, Fernando Ordóñez, Christopher Portway, Milind Tambe, and Craig Western. Using game theory for Los Angeles airport security. *AI Mag.*, 30(1):43–57, 2009.
14. Tim Roughgarden. Stackelberg scheduling strategies. In *STOC*, pages 104–113, New York, NY, USA, 2001. ACM.
15. Heinrich von Stackelberg. *Marktform und Gleichgewicht*. Springer, Vienna, 1934.
16. Bernhard von Stengel and Shmuel Zamir. Leadership with commitment to mixed strategies. Research Report LSE-CDAM-2004-01, London School of Economics, February 2004.
17. David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory of Computing*, 3(1):103–128, 2007.

APPENDIX

A Omitted proofs from Section 3

Lemma 2. *Given overestimates $P_{a'_f}$ and $P_{a''_f}$ on the regions corresponding to a'_f and a''_f, and points $q_{a'_f}$ and $q_{a''_f}$ in these respective regions,* FIND POINT *will either give a point on a new hyperplane for one of the regions $P_{a'_f}$ or $P_{a''_f}$, or will return that $P_{a'_f}$ and $P_{a''_f}$ already have zero intersection volume. This requires at most $O(L)$ samples.*

Proof. If no interior point is found, then the intersection volume must be zero. Now consider the more interesting case when a point p^* is found. Let a^*_f be the follower action produced by this point. There are three possibilities: either $a^*_f = a'_f$, $a^*_f = a''_f$, or a^*_f is not equal to either. Let us consider the first case, where $a^*_f = a'_f$. If we consider the line segment between p^* and $q_{a''_f}$, it is clear that this line segment will intersect with a currently unknown hyperplane for region a''_f. This is because we know that such a hyperplane exists, as a''_f is preferred at point $q_{a''_f}$ and it is not at point p^*. We know that this hyperplane was previously unknown, because when we determined p^*, we made sure that $p^* \in P_{a''_f}$. We can find the point of intersection with binary search, using $O(L)$ samples. The same argument holds true for the other two cases, using the point $q_{a'_f}$ instead of $q_{a''_f}$.

Theorem 1. *To find, w.h.p., all the hyperplanes that separate regions, SU requires $O(Fk \log k + dk^2 L))$ samples, where $F = 1/\epsilon$, ϵ is the smallest volume of regions that we consider, L is the precision, and $k = |A_f|$. Computationally, this requires the solution of $O(k^2)$ linear programs.*

Proof. The first step in SU is to find one point in each region (with sufficient volume). This can be handled by random sampling, as shown by Lemma 1. After we have generated one point in each region, sort the regions by their corresponding follower actions $\{a^1_f, a^2_f ... a^{k'}_f\}$, where $k' \leq k$.

Next, we iterate over all pairs of regions, that is, we iterate over all pairs a'_f and $a''_f \in \{a^1_f, a^2_f ... a^{k'}_f\}$, where $a'_f \neq a''_f$. We run FIND POINT on a'_f and a''_f. If it returns failure, we move on to the next pair of regions. If FIND POINT does find a point, we run DETERMINE HYPERPLANE to find a new hyperplane, after which we update the overestimates.

Since we know that the d-simplex of all leader strategies is composed of at most k convex regions, there are at most $\binom{k}{2}$ separating hyperplanes (as each region can share at most one hyperplane with each other region). It takes $O(L)$ samples per hyperplane to find a single point on it with FIND POINT, according to Lemma 2. It then takes an additional $O(dL)$ samples to find d linearly independent points, according to Lemma 3. Thus, we require $O(dL)$ samples per hyperplane, or $O(k^2 dL)$ total samples to find all of the hyperplanes. This is in addition to the $O(Fk \log k)$ samples necessary to find points in the regions with sufficiently high volume, according to Lemma 1.

Computationally, we need to run at most $\binom{k}{2}$ linear programs that find a valid starting point to determine a hyperplane, since there are at most $\binom{k}{2}$ hyperplanes. In addition, we need to run at most $\binom{k}{2}$ linear programs that fail to find a feasible point, because once we fail to find a feasible point we need never try that pair of follower actions again. This gives us a bound of $O(k^2)$ on the number of linear programs.

B Multiple types/unknown payoffs

In this subsection, we extend the work of Section 3 to the Bayesian case, where there is a set Θ of opponent types ($|\Theta| = \tau$), and the follower's payoff function remains unknown.

First, let us consider a simplified version of this problem, where a sample tells us what *every* type would play for this mixed strategy, instead of what a single type would play. We call such a powerful sample a *complete sample*.

Lemma 4. *The leader can find all the necessary hyperplanes using $O(\tau(kF\log(k) + dk^2L)))$ complete samples.*

Proof. Number the types $\theta^1, \theta^2, ..., \theta^\tau$. We simply run the SU algorithm τ times, first for θ^1, etc. In each case, we ignore all the information except for the type we are currently considering. In the end, we will know all the hyperplanes for each type. $O(Fk\log(k) + dk^2L))$ samples are sufficient to solve the problem with a single type, which gives us the desired bound.

Now let us consider the original problem where in a single sample, we only obtain a type θ drawn according to the distribution, and the action played by that type. We can use the algorithm from Lemma 4 by sampling the same point sufficiently often, so that we obtain a complete sample with all the types—in fact, we only need the type that the algorithm is currently considering.

Theorem 6. *To find all of the hyperplanes requires*
$O(P(\theta')^{-1}\tau(Fk\log(k) + dk^2L)\log(\tau(Fk\log(k) + dk^2L))$ *samples for a constant chance of success, where* $\theta' = \underset{\theta\in\Theta}{\mathrm{argmin}} P(\theta)$

Proof. First, let $z = Fk\log(k)+dk^2L$. Assume that we sample at each point $(P(\theta')^{-1})* \ln(z\tau + 1)$ times. Since we fail a sample with probability $(1 - P(\theta'))$, we can upper bound the chance of failing $(P(\theta')^{-1}) * \ln(z\tau + 1)$ consecutive times as $((1 - \frac{1}{P(\theta')^{-1}})^{P(\theta')^{-1}}))^{\ln(z\tau+1)} < (\frac{1}{e})^{\ln(z\tau+1)} = \frac{1}{(z\tau+1)}$. This gives a lower bound of $(1 - \frac{1}{(z\tau+1)})$ chance of success at a single point. Then, our chance of succeeding at all $z\tau$ distinct points as $(1 - \frac{1}{(z\tau+1)})^{z\tau} > \frac{1}{e}$. Thus we have a chance of success of greater than $\frac{1}{e}$.

Once we have all the hyperplanes, we have enough information to solve for the optimal strategy. To solve this exactly still requires us to solve an NP-hard problem, for example using the MIP from Appendix F.

C Known payoffs/unknown type distribution

In this subsection, we study a different version of the Stackelberg learning problem: we assume that the leader knows the payoff matrix for every follower type, but does not know the (fixed) distribution over types. In each round, the leader commits to a mixed strategy; the follower type is drawn according to the distribution over types; and finally, the follower plays his best response to the mixed strategy given his type. Unlike in Appendix B, the leader only learns the action that the follower plays, not the type. This will allow her to conclude that the follower's type must have been one of a subset of the types, but in general she will not know the type exactly, because multiple types may be consistent with the follower's action. If the leader learns the exact distribution over follower types, then of course she can compute the optimal strategy; however, she may also be able to learn the optimal strategy without learning the exact distribution over types. In fact, in some cases it is not possible to learn the exact distribution—for example, if there are two types for which the optimal response to *any* leader strategy is column 1. The leader's goal is to learn the optimal strategy in as few rounds as possible. We try to minimize the worst-case number of rounds required.

First, we assume that the distribution is degenerate: the follower always has the same type θ, but the leader initially does not know which one. We obtain the following simple result:

Proposition 1 *If the follower has a fixed type θ, then the leader can learn an optimal Stackelberg strategy in τ rounds. Computationally, this requires only polynomial time.*

Proof. Let $\sigma^{\theta'}$ be an optimal Stackelberg strategy for the leader if the follower type is θ' (which can be computed in polynomial time using linear programming). If $\theta_1, \ldots, \theta_\tau$ is an ordering of the types, then in round i, let the leader commit to σ^{θ_i}. In round i, the leader obtains some utility U_l^i. Let $i^* \in \arg\max_{i \in \{1,\ldots,\tau\}} U_l^i$ be a round in which the leader obtained maximal utility. Then, $\sigma^{\theta_{i^*}}$ is an optimal Stackelberg strategy. This is because in the round i such that $\theta_i = \theta$ (the true follower type), the leader will obtain the maximum possible utility.

We now move on to the case of an arbitrary (fixed) distribution over types. Any given leader strategy will result in a distribution over follower actions: given leader strategy σ, the probability that follower action a_f is played is $P(a_f|\sigma) = \sum_{\theta \in \Theta} x_{\theta,\sigma,a_f}$ where x_{θ,σ,a_f} is 1 if a follower of type θ would respond to σ with action a_f, and 0 otherwise. If the leader commits to the same mixed strategy σ for a sufficiently large number of rounds, the leader will (approximately) learn $P(a_f|\sigma)$ for all a_f. (In practice, it may also be desirable not to have to switch strategies too often.) We assume that the leader learns in this manner, that is, by using the same mixed strategy for sufficiently long to learn the $P(a_f|\sigma)$ before switching to another mixed strategy. We call such a period in which the leader only plays a single mixed strategy an *extended sample*. Our objective is to minimize the number of extended samples needed to learn the optimal strategy.

Theorem 7. *If the follower types are drawn independently from a fixed distribution, the leader can learn enough about the distribution to determine an optimal Stackelberg strategy using 2τ extended samples. Computationally, this requires polynomial time.*

Proof. Let A_f^θ be the set of all follower actions that a follower of type θ will play against some leader strategy σ, that is, $A_f^\theta = \{a_f \in A_f : (\exists \sigma)\ BR(\theta, \sigma) = a_f\}$. We can find these sets as follows. $a_f \in A_f^\theta$ if there is a feasible solution to the following set of linear inequalities (where $\sigma(a_l)$ is the probability that the leader puts on a_l):

$$\boxed{\begin{array}{l} (\forall a'_f)\ \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a_f) \geq \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a'_f) \\ \sum_{a_l} \sigma(a_l) = 1 \\ (\forall a_l)\ \sigma(a_l) \geq 0 \end{array}}$$

While these inequalities only guarantee that there is a leader mixed strategy for which a_f is *one* of the best responses, by the nondegeneracy and bounded minimum volume assumptions, there is such a strategy for which a_f is the *unique* best response.

In the learning process, we first determine, for each type θ with $|A_f^\theta| > 1$, the probability $P(\theta)$ of that type. To do so, we find two leader strategies $\sigma^{\theta,1}, \sigma^{\theta,2}$ such that $BR(\theta, \sigma^{\theta,1}) \neq BR(\theta, \sigma^{\theta,2})$, but for any $\theta' \neq \theta$, $BR(\theta', \sigma^{\theta,1}) = BR(\theta', \sigma^{\theta,2})$. Once we have found such a pair of leader strategies, we can extended-sample both of them; as we switch from one to the other, some probability mass will shift from one follower action to another, and the amount of mass must be exactly $P(\theta)$. We find these two points by finding one of the separating hyperplanes for θ, identifying two points (leader mixed strategies) close to but on opposite sides of the hyperplane, and checking that they have the desired properties (if not, we can repeatedly draw a new pair of points in a way that guarantees we will eventually succeed). All of this can be done in polynomial time using linear programming, as we explain next.

First, we need to find two actions $a_f^1, a_f^2 \in A_f^\theta$ that correspond to bordering regions—that is, for which there is a leader mixed strategy such that a follower of type θ is indifferent between these two actions (and strictly prefers them to all other actions). We let a_f^1 be an arbitrary member of A_f^θ. Then, for every $a'_f \in A_f^\theta$ with $a'_f \neq a_f^1$, we check if it can take the role of a_f^2, by solving the following linear program:

$$\boxed{\begin{array}{l} \textbf{maximize } \epsilon \\ \textbf{subject to} \\ \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a_f^1) = \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a'_f) \\ \\ (\forall a''_f \notin \{a_f^1, a'_f\})\ \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a_f) \geq \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a''_f) + \epsilon \\ \\ \displaystyle\sum_{a_l} \sigma(a_l) = 1 \\ (\forall a_l)\ \sigma(a_l) \geq 0 \end{array}}$$

If the optimal solution has a positive objective value, then it corresponds to a mixed strategy σ such that the follower is indifferent between a_f^1 and a_f^2 $(= a'_f)$, and strictly prefers these two actions to all other actions. Now, we can find two points σ^1 and σ^2, each within distance δ of σ, such that the follower strictly prefers to play a_f^1 against σ^1, and strictly prefers to play a_f^2 against σ^2. We can then check whether these two points satisfy the required conditions for $\sigma^{\theta,1}$ and $\sigma^{\theta,2}$; if they do not, we can find points that do by repeatedly shrinking δ and perturbing σ on its hyperplane (which is guaranteed to work due to the nondegeneracy/minimum volume assumptions).

Hence, we can find $P(\theta)$ for every θ with $|A_f^\theta| > 1$. After we have done so, for the θ with $|A_f^\theta| = 1$ (the types that respond to every leader strategy with the same action), we cannot learn their individual probabilities, as pointed out before; however, all that is needed to find an optimal leader strategy is, for each action a_f, the total probability of the types that always play a_f. We can infer all these probabilities from any single extended sample, as follows. We already know $P(\theta)$ for every θ with $|A_f^\theta| > 1$, and we know which actions these types play at the extended sample. So, we can subtract these probabilities from the action probabilities in the extended sample, and the remaining probabilities on actions are the probabilities that we want.

For each type, we used at most 2 extended samples, resulting in at most 2τ extended samples. Computationally, this approach requires solving at most $2\tau|A_f|$ linear programs.

This will discover almost the entire distribution, with one exception: if there are two (or more) types that always play the same action, then it is impossible to distinguish them and we can only learn their aggregate probability. By the nondegeneracy assumption, this can only happen if there is a fixed follower action that is the best response for those types against *any* leader strategy—that is, there are no separating hyperplanes for those types. Of course, knowing the aggregate probability is sufficient for computing the optimal Stackelberg strategy for the leader, because if two types are indistinguishable then we may as well merge them into a single type. Of course, even given the distribution, it is NP-hard to compute the optimal mixed strategy in this context. This is not in contradiction with the above theorem, which only considers the computation needed to learn enough about the distribution. To find the optimal mixed strategy, an NP-hard problem still needs to be solved, which can be done, for example, using the MIP in Appendix F

D Omitted proofs from Section 4

Theorem 2. *For all constant $\epsilon > 0$, no polynomial-time factor-$\tau^{1-\epsilon}$ approximation exists for BOFT unless NP $=$ P, even if there are only two follower actions.*

Proof. It is known that no polynomial-time factor-$|V|^{1-\epsilon}$ approximation exists for MAX-INDEPENDENT-SET (given by a graph (V, E)), unless NP = P [17]. We show our result by reducing an arbitrary instance (V, E) of this problem to a game as follows. For every $v \in V$, there is a follower type θ^v, and a leader action a_l^v. The prior over follower types is uniform. There are two follower actions, A and B (for each follower type). The leader gets utility 1 if the follower plays A, and 0 otherwise. The follower's utility is defined as follows.

- For all $v \in V$, $u_f(\theta^v, a_l^v, A) = |V|$.
- For all $v, w \in V$ with $v \neq w$, $u_f(\theta^v, a_l^w, A) = 0$.
- For all $(v, w) \in E$, $u_f(\theta^v, a_l^w, B) = 1 + |V|^2$
- For all $(v, w) \notin E$, $u_f(\theta^v, a_l^w, B) = 1$.

Suppose there is an independent set S of size k in (V, E). Consider a mixed strategy that places probability $\frac{1}{k}$ on each a_l^v with $v \in S$. Then, for every type θ^v with $v \in S$, the follower will play A, because the follower will get $\frac{n}{k} \geq 1$ for playing A, and 1 for playing B (because no a_l^w with $(v, w) \in E$ is ever played, because S is an independent set).

Correspondingly, suppose there is a leader strategy that gets k types to play A. Let S be the set of vertices v such that the follower plays A for θ^v; we will show it is an independent set. If θ^v plays A, then a_l^v must get probability at least $1/n$ (to make playing A optimal for the follower). But, no action a_l^w with $(v, w) \in E$ can get probability at least $\frac{1}{n}$, because in that case the expected utility for the follower (with type θ^v) of playing B is at least $(\frac{1}{n})(1 + n^2) > n$. So the k types must constitute an independent set.

Hence, we have shown that the number of types playing A (which is proportional to the leader's utility) for the optimal leader strategy is equal to the size of the maximum independent set. Since the number τ of types for the follower is equal to $|V|$, this gives us the desired result.

Theorem 3. *There is a polynomial-time factor-τ approximation algorithm for BOFT.*

Proof. Let σ^* be an optimal leader strategy, that is,
$\sigma^* \in \arg\max_\sigma \sum_{\theta \in \Theta} P(\theta) \sum_{a_l} \sigma(a_l) u_l(a_l, BR_f(\theta, \sigma))$. Consider the following simple randomized algorithm: choose a type θ uniformly at random and play a mixed leader strategy σ^θ that maximizes utility against that single type, that is,
$\sigma^\theta \in \arg\max_\sigma \sum_{a_l} \sigma(a_l) u_l(a_l, BR_f(\theta, \sigma))$. (We can find such a mixed leader strategy in polynomial time by the linear programming approach from [2].) For every θ, we have
$\sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta, \sigma^\theta)) \geq \sum_{a_l} \sigma^*(a_l) u_l(a_l, BR_f(\theta, \sigma^*))$. The probability that θ is chosen both by our algorithm and by nature as the type of the follower is $(1/\tau)P(\theta)$. Because utilities are bounded below by zero, the expected utility that we receive is at least $\sum_{\theta \in \Theta} (1/\tau) P(\theta) \sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta, \sigma^\theta)) \geq$
$(1/\tau) \sum_{\theta \in \Theta} P(\theta) \sum_{a_l} \sigma^*(a_l) u_l(a_l, BR_f(\theta, \sigma^*))$. Hence, this randomized algorithm results in a factor-τ approximation. (We emphasize that this algorithm randomly chooses a mixed strategy to commit to, which is not the same as committing to the corresponding mixture of those mixed strategies.)

Instead of randomizing uniformly over which of the σ^θ to commit to, we can instead, for each θ, evaluate the total expected utility that results from committing to the strategy θ (which is
$\sum_{\theta' \in \Theta} P(\theta') \sum_{a_l} \sigma^\theta(a_l) u_l(a_l, BR_f(\theta', \sigma^\theta))$), and choose one that maximizes this expected utility. This cannot lead to a lower expected utility for the leader; hence, this gives us a deterministic algorithm with the same approximation guarantee.

Theorem 4. *WOFT is completely inapproximable in polynomial time, unless P=NP (that is, it is hard to distinguish between instances where the leader can get at least* 1 *in the worst case, and instances where the leader can only get* 0*)—even if there are only four follower actions.*

Proof. We reduce an arbitrary instance of 3SAT to a game such that the leader can obtain an expected utility of 1 if the 3SAT instance is satisfiable, and 0 otherwise. The

3SAT instance consists of n variables, $x_1, \ldots, x_n$, and m clauses, $C_1, \ldots, C_m$. We create one type for each variable and for each clause. In the game, for every variable x_i, we have two leader actions, $a_l^{+x_i}$ and $a_l^{-x_i}$. The follower has four actions, A, B, C, and D. The leader gets utility 0 if the follower plays A, and 1 otherwise. There are two kinds of follower type: one for each variable (θ^{x_i}) and one for each clause (θ^{C_j}).

For a type θ^{x_i} corresponding to a variable, we make it so that actions C and D are always suboptimal for the follower, so we only consider A and B. We let $u_f(\theta^{x_i}, a_l^{+x_i}, A) = u_f(\theta^{x_i}, a_l^{-x_i}, A) = n$, and $u_f(\theta^{x_i}, a_l^{+x_j}, A) = u_f(\theta^{x_i}, a_l^{-x_j}, A) = 0$ for $i \neq j$. Also, we let $u_f(\theta^{x_i}, a_l, B) = 1$ for all a_l. The following table gives the payoff matrix for type θ^{x_1} as an example.

M_{x_1}:

	A	B
$+x_1$	0,n	1,1
$-x_1$	0,n	1,1
	0,0	1,1
	0,0	1,1
	.	.
	.	.
	.	.

For a type θ^{C_j} corresponding to a clause, for each literal λ in the clause (where the set of all literals is $\{+x_i : i \in \{1, \ldots, n\}\} \cup \{-x_i : i \in \{1, \ldots, n\}\}$), exactly one of the three actions B, C, D will give the follower utility n if the leader plays a_l^{λ} (and each of these three actions will correspond to one of the three literals in the clause). Playing A always gives the follower utility 1. Otherwise, the follower gets 0. For example, if $C_1 = (+x_1 \vee -x_2 \vee +x_4)$, then the following table gives the payoff matrix for type θ^{C_1}.

M_{c_1}:

	A	B	C	D
$+x_1$	0,1	1,n	1,0	1,0
$-x_1$	0,1	1,0	1,0	1,0
$+x_2$	0,1	1,0	1,0	1,0
$-x_2$	0,1	1,0	1,n	1,0
$+x_3$	0,1	1,0	1,0	1,0
$-x_3$	0,1	1,0	1,0	1,0
$+x_4$	0,1	1,0	1,0	1,n
$-x_4$	0,1	1,0	1,0	1,0

We now show that the leader can obtain a utility of 1 in this game against every type if and only if the 3SAT instance has a satisfying assignment (and will get 0 against at least one type otherwise). Let σ be the mixed strategy to which the leader commits.

For each variable x_i, if $\sigma(a_l^{+x_i}) + \sigma(a_l^{-x_i}) \leq \frac{1}{n}$, then a follower of type θ^{x_i} will play B, otherwise it will play A. Hence, the leader will get 1 for all types corresponding to variables if and only if the above inequality holds for every variable x_i; otherwise, the leader will obtain 0 against at least one type corresponding to a variable.

For each clause C_j, if for at least one of the three literals λ in the clause, we have $\sigma(a_l^{\lambda}) \geq \frac{1}{n}$, then a follower of type θ^{C_j} will play B, C, or D; otherwise, it will play A.

Hence, the leader will get 1 for all types corresponding to clauses if and only if every clause contains at least one literal for which the above inequality holds; otherwise, the leader will obtain 0 against at least one type corresponding to a clause.

Now, suppose that the 3SAT instance has a satisfying assignment. Then, consider the mixed strategy that places probability $1/n$ on every literal that is set to *true* in the satisfying assignment. For every variable x_i, we have $\sigma(a_l^{+x_i}) + \sigma(a_l^{-x_i}) \leq \frac{1}{n}$, so the follower will play B for all the types corresponding to variables. For every clause, for at least one of the three literals λ in the clause, we have $\sigma(a_l^{\lambda}) \geq \frac{1}{n}$ (because the assignment satisfies the formula), so the follower will play B, C, or D for all the types corresponding to clauses. Hence, the leader obtains utility 1 for every follower type.

Conversely, suppose that there exists a mixed strategy such that the leader obtains positive utility for every follower type. For every variable x_i, we must have $\sigma(a_l^{+x_i}) + \sigma(a_l^{-x_i}) \leq \frac{1}{n}$. Hence, at most one of $a_l^{+x_i}$ and $a_l^{-x_i}$ can receive probability at least $1/n$. Now consider the following assignment: if $a_l^{+x_i}$ receives probability at least $1/n$, set x_i to *true*; if $a_l^{-x_i}$ receives probability at least $1/n$, set x_i to *false*; otherwise, set x_i arbitrarily. Because the leader receives utility at least 1 against every type corresponding to a clause, for every clause, for at least one of the three literals λ in the clause, we must have $\sigma(a_l^{\lambda}) \geq \frac{1}{n}$. But that means that the clause either contains some $+x_i$ where x_i is set to *true*, or some $-x_i$ where x_i is set to *false*. It follows that our assignment is a satisfying assignment.

Theorem 5. *WORT is NP-hard.*

Proof. We reduce an arbitrary instance of 3-COVER (where we are given a set of elements S ($|S| = n$), a collection of subsets $S_i \subseteq S$ with $|S_i| = 3$, and we are asked whether all of S can be covered with $n/3$ of the S_i) to a game with ranges where the leader can obtain an expected utility of at least $3/n$ if and only if a 3-cover exists.

For each S_i, let there be both a row $a_l^{S_i}$ and a column $a_f^{S_i}$. Also, let there be a column a_f^s for each s (these columns are really bad for the leader and must be avoided). Let the utilities be defined as follows:
$u_l(a_l^{S_i}, a_f^{S_i}) = 1$
$u_l(a_l^{S_i}, a_f^{S_j}) = 0$ for $i \neq j$
$u_f(a_l^{S_i}, a_f^{S_i}) \in [0, 1]$
$u_f(a_l^{S_i}, a_f^{S_j}) = 0$ for $i \neq j$
$u_f(a_l^{S_i}, a_f^{s}) = 1$ if $s \notin S_i$
$u_f(a_l^{S_i}, a_f^{s}) = -n$ if $s \in S_i$

If there exists a 3-cover (of size $n/3$), then the leader can obtain guaranteed utility $3/n$, as follows: randomize uniformly over the strategies corresponding to the 3-cover (probability $3/n$ each). The follower will not be incentivized to play any a_f^s, because that gives him an expected utility of at most $-3 + 1 = -2$. The follower will not be incentivized to play an $a_f^{S_j}$ for which S_j is not in the 3-cover. because it will give him utility 0 (note that ties are broken in favor of the leader, as always). The follower may be incentivized to play any $a_f^{S_i}$ for which S_i in the 3-cover; for each of these, the leader will get $3/n$ in expectation.

Conversely, if the leader can get guaranteed utility $3/n$, consider the set of all the S_i for which $a_l^{S_i}$ receives positive probability for the leader. The claim is that this must be a 3-cover (of size $3/n$). First, the follower cannot be incentivized to play any a_f^s. Hence, for each s, some $a_l^{S_i}$ with $s \in S_i$ must get positive probability for the leader. Now suppose strictly more than $n/3$ of the $a_l^{S_i}$ get positive probability for the leader. Then one of them must get probability less than $3/n$. The column player might be incentivized to play the corresponding $a_f^{S_i}$ (since that may be the only one that ever gives the follower positive utility), in which case the row player's expectation is less than $3/n$, contrary to assumption.

E Worst-case optimization for linked range types

We now define a generalization of WORT (from subsection 4.3, which we can prove is inapproximable unless $P = NP$. This generalization allows the follower's payoffs to be linked across entries. We refer to this problem as *worst-case optimization for linked range types (WOLRT)*. Specifically, instead of having ranges for each follower entry, we now have a linear expression for each follower entry which may involve *symbols*: an example expression would be $3c_1 + 4c_2 + 1$. For each symbol, there is a range (for example, $c_1 \in [0, 1]$—in fact, without loss of generality, we can assume every range is $[0, 1]$), and a symbol can occur in multiple entries.

For example, consider the following game with linked ranges, with $c_1, c_2 \in [0, 1]$:

	L	R
U	$0, 1 + c_1$	$1, 0$
D	$1, 0$	$0, 1 + c_1/2 + c_2/2$

If the leader places less than $3/7$ probability on U, then the follower is guaranteed to play R; this results in a utility of at most $3/7$ for the leader. If the leader places more than $3/5$ probability on U, then the follower is guaranteed to play L; this results in a utility of at most $2/5$ for the leader. If the leader places probability between $3/7$ and $3/5$ on U, then the follower may end up playing either L or R; by placing probability $1/2$ on U, the leader obtains an expected utility of $1/2$, which is optimal.

We note that WOLRT generalizes WORT, because we can have a separate symbol for every entry.

Theorem 8. *WOLRT is completely inapproximable in polynomial time, unless P=NP (that is, it is hard to distinguish between instances where the leader can get at least* 1 *in the worst case, and instances where the leader can only get* 0*).*

Proof. We reduce an arbitrary SET-COVER instance (where we are given a set S, a collection of subsets S_i of S, and a number k, and are asked whether all of S can be covered with at most k of the S_i) to a WOLRT instance such that the leader can get utility 1 in the worst case if there is a set cover of size at most k, and 0 otherwise.

With every $s \in S$, we associate a symbol $c_s \in [0, 1]$. For every S_i, the leader has an action $a_l^{S_i}$, and the follower has an action $a_f^{S_i}$. Additionally, for every $s \in S$, the follower has an action a_f^s. We have:

$u_l(\cdot, a_f^s) = 0$
$u_l(\cdot, a_f^{S_i}) = 1$
$u_f(\cdot, a_f^s) = c_s$
$u_f(a_l^{S_i}, a_f^{S_i}) = k \sum_{s \in S_i} c_s$
$u_f(a_l^{S_i}, a_f^{S_j}) = 0$ for $i \neq j$

If there is a covering of size k, then the leader can uniformly randomize over the $a_l^{S_i}$ corresponding to that covering. Then, for any $s' \in S$, if the follower plays one of the $a_f^{S_i}$ where S_i is in the covering and $s' \in S_i$, his expected utility is at least $(1/k) \cdot k(\sum_{sinS_i} c_s) \geq c_{s'}$; so there is no reason for the follower to play $a_f^{s'}$, and the leader is guaranteed a utility of 1.

Conversely, suppose that there is a strategy σ that guarantees the leader positive utility, that is, it guarantees that the follower will play one of the $a_f^{S_i}$. For any $s' \in S$, consider the scenario where $c_{s'} = 1$ and the other c_s are 0. The follower is incentivized to play some $a_f^{S_i}$; it must be that $s' \in S_i$, and the follower's expected payoff for playing this is $\sigma(a_l^{S_i}) \cdot k \sum_{sinS_i} c_s) = \sigma(a_l^{S_i}) \cdot k$, so it follows that $\sigma(a_l^{S_i}) \geq 1/k$. There can be at most k subsets S_i for which this is true, and they must cover all the $s' \in S$, so it follows there is a covering of size at most k.

F Mixed integer program formulations of BOFT and WOFT

First let us introduce a mixed integer program (which, in our view, simplifies the known mixed integer program [12] slightly, but the idea is similar). It uses auxiliary variables $q(\theta, a_l, a_f)$, which correspond to the probability that a_l, a_f are played, given that the follower has type θ—which will be equal to 0 if a_f is not a best response for θ, and equal to $\sigma(a_l)$ otherwise. It also uses binary indicator variables $b(\theta, a_f) \in \{0, 1\}$ for whether the best response for type θ is a_f.

maximize $\sum_\theta P(\theta) \sum_{a_l, a_f} q(\theta, a_l, a_f) u_l(a_l, a_f)$
subject to
$(\forall \theta)\ \sum_{a_f} b(\theta, a_f) = 1$
$(\forall \theta, a_l, a_f)\ q(\theta, a_l, a_f) \leq b(\theta, a_f)$
$(\forall \theta, a_l)\ \sum_{a_f} q(\theta, a_l, a_f) = \sigma(a_l)$
$(\forall \theta, a_f, a_f')\ \sum_{a_l} q(\theta, a_l, a_f)(u_f(\theta, a_l, a_f) - u_f(\theta, a_l, a_f')) \geq 0$
$\sum_{a_l} \sigma(a_l) = 1$
$(\forall a_l)\ \sigma(a_l) \geq 0$

The following is a mixed integer program (MIP) formulation for WOFT. In this MIP, we assume that all payoffs are normalized to lie in $[0, 1]$. Again, we use a binary variable $b(\theta, a_f) \in \{0, 1\}$ that indicates whether a_f is the best response for θ. We also use variables U_l (the leader's worst-case utility), $U_l(\theta)$ (the leader's utility if the type is θ), $U_f(\theta, a_f)$ (the follower's utility for playing a_f given θ), $U_f'(\theta, a_f)$ (equal to $U_f(\theta, a_f)$ if a_f is the best response for θ, 0 otherwise), $U_f(\theta)$ (the follower's utility if the type is θ).

maximize U_l

subject to

$(\forall \theta)\ \sum_{a_f} b(\theta, a_f) = 1$

$(\forall \theta, a_f)\ U_f(\theta, a_f) = \sum_{a_l} \sigma(a_l) u_f(\theta, a_l, a_f)$

$(\forall \theta, a_f)\ U'_f(\theta, a_f) \leq U_f(\theta, a_f)$

$(\forall \theta, a_f)\ U'_f(\theta, a_f) \leq b(\theta, a_f)$

$(\forall \theta)\ U_f(\theta) = \sum_{a_f} U'_f(\theta, a_f)$

$(\forall \theta, a_f)\ U_f(\theta) \geq U_f(\theta, a_f)$

$(\forall \theta, a_f)\ U_l(\theta) \leq \sum_{a_l} \sigma(a_l) u_l(a_l, a_f) + (1 - b(\theta, a_f))$

$(\forall \theta)\ U_l \leq U_l(\theta)$

$\sum_{a_l} \sigma(a_l) = 1$

$(\forall a_l)\ \sigma(a_l) \geq 0$