Learning Optimal Strategies to Commit To[*]

Binghui Peng[1] and **Weiran Shen**[1] and **Pingzhong Tang**[1] and **Song Zuo**[2]
[1]Institute for Interdisciplinary Information Sciences, Tsinghua University
[2]Google Research
pbh15@mails.tsinghua.edu.cn, emersonswr, kenshinping@gmail.com, szuo@google.com

Abstract

Over the past decades, various theories and algorithms have been developed under the framework of Stackelberg games and part of these innovations have been fielded under the scenarios of national security defenses and wildlife protections. However, one of the remaining difficulties in the literature is that most of theoretical works assume full information of the payoff matrices, while in applications, the leader often has no prior knowledge about the follower's payoff matrix, but may gain information about the follower's utility function through repeated interactions. In this paper, we study the problem of learning the optimal leader strategy in Stackelberg (security) games and develop novel algorithms as well as new hardness results.

Introduction

Computing solution concepts in game theory has been one of the most important research agendas at the interface of computer science and economics. Over the past twenty years, various solution concepts have been investigated through the lens of computation, such as Nash equilibrium (Conitzer and Sandholm 2008; Chen, Deng, and Teng 2009; Daskalakis, Goldberg, and Papadimitriou 2009; Chen et al. 2017; Chen, Tang, and Wang 2017), Stackelberg equilibrium (Conitzer and Sandholm 2006; Letchford and Conitzer 2010; Zuo and Tang 2015), correlated equilibrium (Papadimitriou and Roughgarden 2008; Jiang and Leyton-Brown 2011).

Among these solution concepts, of particular interest to the AI community is the computation of Stackelberg equilibrium, also known as *the optimal strategy to commit to*, for several well-known successful applications of this solution concept to the domain of security, established by the pioneer work of Tambe (2011).

In a two-player normal-form game, a commitment is a mixed strategy announced by one player, called the leader. The other player, called the follower, responds to the leader's commitment according to its rationality model and both players derive payoffs. The problem is to find the leader's best commitment.

When the follower is perfectly rational, i.e., choosing a pure strategy that maximizes its payoff given the commitment, the algorithmic problem has been nicely solved by Conitzer and Sandholm (2006). The idea is to solve a sequence of LPs (that optimizes leader's commitment), one for each follower's pure strategy, constrained on that strategy being the best response for the follower.

The above algorithm has been adapted to a number of realistic applications, for example, the LAX airport security resource allocation (Pita et al. 2008), Boston coastal protection (Jain et al. 2010), and other security game applications (An et al. 2011; Xu et al. 2015; Yin et al. 2015). However, there are a number of limitations that prevent the algorithm from being further applied. One of such serious limitations is that in real applications, the payoff matrices are rarely given as inputs (Camerer, Ho, and Chong 2004).

This paper aims to resolve this issue. In our setting, the leader has no prior knowledge about the follower's payoff matrix, yet still tries to figure out the optimal strategy to commit to. In order to do so, the leader could gain knowledge about the follower's utility function from interactions with the follower. We summarize this interaction as a *response oracle*, where the leader can commit to a mixed strategy and get the best response of the follower via a sample of response oracle. We believe that the response oracle is rare and valuable in real life and our goal is to minimize the number of samples needed to identify the optimal commitment.

Our Contributions

Prior to our work, Letchford, Conitzer, and Munagala (2009) first study this problem and give a sampling algorithm for this problem. However, their algorithm works well only when the smallest volume of the feasible region for the follower's action is large, otherwise, their algorithm would be no better than brute force search in the worst case. To be more specific, the sample complexity in Letchford, Conitzer, and Munagala (2009) is $O(V^{-1}n\log n + mn^2L)$, where m, n, L, V is the number of the leader's and the follower's actions, the representation precision and the volume of the smallest feasible region, respectively. We note that the term V^{-1} can be of $O(2^{mL})$, with exponential dependence on both L and m in the worst case and their algorithm is inef-

[*]The authors thank the anonymous reviewers for their helpful comments. This paper is supported in part by the National Natural Science Foundation of China Grant 61561146398, a China Youth 1000-talent Program, and an Alibaba Innovative Research Program.

ficient even when the leader only has two actions. The main inefficiency comes from their procedure for identifying all *effective* actions for the follower. We aim to tackle this problem in this paper and we develop novel algorithms as well as hardness results regarding the sample complexity. Our algorithm is quite general and flexible. With some extensions, it also applies to other Stackelberg games, e.g., the security game. To summarize, we make the following contributions in this paper.

- For general Stackelberg games, we propose an algorithm, named SEPARATE-SEARCH, whose sample complexity is $\mathrm{poly}(m, n, L)\mathrm{Ext}(A)$. $\mathrm{Ext}(A)$ is the number of extreme points which is independent of L. Thus,we can get rid of the exponential dependence on L. Moreover, when m is a constant, the sample complexity is polynomial in L and n, the same holds for n, i.e., when n is a constant, it is polynomial in L and m. Thus when m or n is constant, our algorithm can be considered to be efficient. However, we note that when $n = \Theta(m)$, $\mathrm{Ext}(A)$ is still $O(\exp(m))$.
- We derive hardness results which show that the exponential dependence of m is inevitable. More specifically, we show that there exists an instance such that any algorithm must take at least $\Omega(\exp(m))$ samples to compute the optimal strategy. Furthermore, the lower bound construction indicates that $\mathrm{Ext}(A)$ many queries are necessary in that case and thus our algorithm is tight.
- We point out several applications of SEPARATE-SEARCH. Especially, in the application of security games, an extension of SEPARATE-SEARCH only requires $O(n^3L)$ samples to identify the optimal strategy to commit, which improves the $\tilde{O}(n^{6.5}L)$[1] sample complexity by Blum, Haghtalab, and Procaccia (2014). Furthermore, our algorithm is much simpler than their algorithm.

Related Work

Our work is most relevant to Letchford, Conitzer, and Munagala (2009), where the authors develop the first sampling algorithm for learning the optimal strategy in Stackelberg games. However, as pointed out in the previous section, their algorithm is highly inefficient when some feasible regions have small volumes. The other most relevant prior work is Blum, Haghtalab, and Procaccia (2014), where the authors give the first efficient sampling algorithm for security games. Their approach is different from ours and they invoke an optimization method from Kalai and Vempala (2006). More specifically, they need to optimize a linear function over a convex region, when only an initial point and the membership query are given. The optimization method in Kalai and Vempala (2006) is quite powerful and elegant, but it is inevitably costly.

There are a bunch of other prior works (Balcan et al. 2015; Marecki, Tesauro, and Segal 2012; Haghtalab et al. 2016; Sinha, Kar, and Tambe 2016; Blum, Haghtalab, and Procaccia 2015) that intend to provide theoretical guarantees regarding learning problems in Stackelberg (security) games under different settings. Balcan et al. (2015) study the problem in the context of no-regret learning, where the defender is not playing with the same fixed attacker, but with an adversary attacker. They compare the online algorithm with the best fixed mixed strategy in hindsight and a low regret bound is derived in their paper.

Haghtalab et al. (2016) and Sinha, Kar, and Tambe (2016) study the problem in the PAC model. In particular, Haghtalab et al. (2016) assume that the attacker has bounded rationality and conclude that polynomial number of adversary responses to three defender strategies that are sufficiently different from each other are sufficient to learn the attacker's utility with high accuracy. The assumption of bounded rationality is crucial since repeating a leader's strategy would allow the algorithm to obtain the utility information for all targets, which does not hold for the perfect rational attacker.

For more details about theoretical learning approaches with regard to the Stackelberg (security) game, we refer readers to the survey of Blum, Haghtalab, and Procaccia (2015).

Another line of works focus on providing practical learning algorithm (Yang et al. 2014; Amin, Singh, and Wellman 2016; Kamra et al. 2018; Ling, Fang, and Kolter 2018; Shen, Tang, and Zuo 2018). Practical learning and optimization algorithms, like deep learning (Goodfellow et al. 2016) and gradient descent, play important roles in these algorithms. Different from our work, most of these work only provide experiment results and are lack of theoretical guarantees.

Preliminary

Stackelberg Games In a standard Stackelberg game, there are two players who choose their strategies one after another. The player who moves first is called the *leader* and the other one is called the *follower*. Formally, we use L and F to denote the sets of pure strategies of the leader and the follower, respectively. Throughout the paper, we only consider finite Stackelberg games, i.e., the sets L and F are both finite. In addition, we use U^{l} and U^{f} to denote the utility functions of the leader and the follower, each of which maps a pair of strategies $(\mathrm{l}, \mathrm{f}) \in \mathrm{L} \times \mathrm{F}$ to a real number.

The strategy chosen by the leader is also called the *commitment* of the leader, which is usually a mixed strategy, i.e., a probabilistic distribution p over the strategy set L. In contrast, the strategy chosen by the follower, called the *response*, is usually a pure strategy. When the follower chooses the response, the leader's commitment p is already chosen, hence the choice of the response won't affect the commitment.

The equilibrium of the game is called *Stackelberg equilibrium*, where the leader commits to the optimal commitment in the sense that its expected utility is maximized if the follower best responds to the commitment. Formally, a pair of commitment and response (p^*, f^*) is a Stackelberg equilibrium, if

$$p^* \in \arg\max_{p \in \Delta\mathrm{L}} \mathrm{E}_{\mathrm{l}\sim p}[U^{\mathrm{l}}(\mathrm{l}, \mathrm{f}^*(p))], \quad \mathrm{f}^* = \mathrm{f}^*(p^*),$$

[1]$\tilde{O}(\cdot)$ omits logarithmic terms of m, n, L

where $\Delta \mathrm{L}$ is the set of the probability distributions over the leader's strategy set L and

$$\mathrm{f}^*(p) = \arg\max_{\mathrm{f}\in\mathrm{F}} \mathrm{E}_{\mathrm{l}\sim p}[U^{\mathrm{f}}(\mathrm{l},\mathrm{f})]$$

is the best response to commitment p.[2] Note that in general, the leader's optimal commitment is not necessarily a best response to the follower's response.

We slightly abuse notation and define $U^{\mathrm{l}}(p,\mathrm{f})$ and $U^{\mathrm{f}}(p,\mathrm{f})$ to be the expected utilities of the leader and the follower under commitment p and response f, i.e.,

$$U^{\mathrm{l}}(p,\mathrm{f}) = \mathrm{E}_{\mathrm{l}\sim p}[U^{\mathrm{l}}(\mathrm{f})], \quad U^{\mathrm{f}}(p,\mathrm{f}) = \mathrm{E}_{\mathrm{l}\sim p}[U^{\mathrm{f}}(\mathrm{l},\mathrm{f})].$$

The standard way of finding a Stackelberg equilibrium (Conitzer and Sandholm 2006) is through solving a series of linear programs for $\forall j \in [n]$ and choose the strategy that maximizes the leader's utility.

$$\begin{aligned} &\max U^{\mathrm{l}}(p,j) \\ \text{s.t.}\quad & U^{\mathrm{f}}(p,j') \le U^{\mathrm{f}}(p,j) \quad \forall j' \ne j, j' \in [n] \\ & \sum_{i=1}^{m} p_i = 1 \\ & p_i \ge 0 \quad \forall i \in [m], \end{aligned}$$

where $n = |\mathrm{F}|$ is the number of the follower's actions and $m = |\mathrm{L}|$ is the number of the leader's actions.

For ease of presentation, we use the following definition:

Definition 1 (Effective actions and feasible regions). *Follower action a_i is an effective action, if there exists a leader commitment p, such that a_i is a best response to p. Also, given a follower's action a_i, the corresponding feasible region, denoted by $\mathcal{P}_i$ is the set of all leader commitments such that a_i is the follower's best response.*

It is easy to see that any two feasible regions are disjoint, and that $\mathcal{P}_j$ is a convex region (may be empty) induced by at most $m+n$ half planes. The separating plane between $\mathcal{P}_i$ and $\mathcal{P}_j$ refers the hyperplane where the utility for playing a_i and a_j are the same for the follower.

Stackelberg Security Games A Stackelberg security game is a special Stackelberg game with some specific structures between the strategy sets and utility functions. More specifically, in the canonical setting of Stackelberg security game, there is a set of targets $T = \{1,\dots,n\}$ and the *defender* (leader) commits to an allocation of *resources* to protect the targets from the *attacker* (follower).

More formally, we use the following standard notation to describe the Stackelberg security game.

- *Resources*. The resources are described by a set R. When there are different types of resources, there is a function $A : R \mapsto 2^{\mathcal{D}}$, where $A(r)$ is the set of *schedules* to which resource r can be assigned.
- *Schedules*. The set of schedules $\mathcal{D}$ is a collection of subsets of targets, i.e., $\mathcal{D} \subseteq 2^T$. For every $D \in \mathcal{D}$, targets in D can be simultaneously protected by one resource. We say t is covered if $t \in D$. Furthermore, we make the assumption that $\mathcal{D}$ is subset close, i.e., if $D \in \mathcal{D}$, then all subsets for D are also contained in $\mathcal{D}$.
- *Utility*. If a target t is attacked, the defender's utility is $U_c^d(t)$ if t is covered, and $U_u^d(t)$ if not. Similarly, the attacker's utility is $U_c^a(t)$ if t is covered, and $U_u^a(t)$ if not. It is commonly assumed that $U_c^d(t) \ge U_u^d(t)$ and $U_c^a(t) \le U_u^a(t)$. We note that it makes no difference to the players' utilities whether a target is covered by one resource or by more than one resources.

A pure strategy for the defender is a valid allocation of resources for schedules, we use Q to denote the set of pure strategies with $q = |Q|$. Let p be the coverage probability of the targets. Define $M_{n\times q}$ to be a binary matrix where $M_{ti} = 1$ if target t is covered in the i-th pure strategy. Then we say p is *implementable* if there exists a mixed strategies s such that $p = Ms$ and $\sum_{i=1}^{q} s_i = 1$.

Response Oracle As we discussed in the introduction, in practice, the leader's knowledge about the follower's utility function U^{f} can be very limited. However, the leader is still able to learn the optimal commitment from interactions.

In this paper, we summarize the experience from interactions as a *response oracle* $\mathcal{R}$. The oracle will output the best response $\mathcal{R}(p)$ when the leader makes a query p (a commitment) to it. The leader then determines the optimal commitment by making queries to the response oracle. In particular, we allow the queries to be adaptive, meaning that the leader could adaptively choose queries based on the results from past queries.

Formally, we make the following assumption about the response oracle $\mathcal{R}$ and utility functions:

- The follower's utility are non-degenerated, i.e., no $(m+1)$ separating planes or boundary planes intersect into one point. Moreover, no separating planes coincide.
- We assume that the game can be specified using limited amount of precision. More precisely, all the entries in the follower utility matrix are in $[-1,1]$ (we can always scale the matrix) and can be presented by L bits. Moreover, the feasible region $\mathcal{P}_i$ for all actions $i \in \mathrm{F}$ has a volume of either 0 (empty) or at least 2^{-nL}.
- When there are more than one best responses for the follower, we can choose any one as we want.

We make the second assumption because if we allow the feasible region of an action $a \in \mathrm{F}$ to be arbitrary small, then it would take infinitely many samples in order to check whether action a is effective.

We focus on the number of such queries made by the leader. We say that a sampling scheme is efficient if the sample complexity is $\mathrm{poly}(m,n,L)$.

Previous Results Before proceeding to our main results, we provide a quick summary of some previous results under

[2]When there are multiple best responses, we follow the convention and break ties in favor of the leader, i.e., the follower always chooses the one that maximizes the leader's expected utility.

this setting. Some of them may serve as subroutines of our algorithm.

The following result comes from (Letchford, Conitzer, and Munagala 2009), it states that for any two actions $a_1, a_2 \in \mathrm{F}$, if we already know a feasible point for each of them, but do not know the separating hyperplane between them. Then with polynomial number of samples, we are able to either find a new separating hyperplane (not necessarily the one between them) or discover a new effective action in F. Formally, we have

Theorem 2. *(Restate, (Letchford, Conitzer, and Munagala 2009)) Given two effective actions a_1 and a_2 for the follower and the corresponding feasible points p_1, p_2, if the separating hyperplane between $\mathcal{P}_1$ and $\mathcal{P}_2$ is still unknown, then with $O(mL)$ samples, we can either find a new separating hyperplane, or discover a new effective action in* F.

We call the algorithm mentioned above FIND-SEPARATING-PLANE. Notice that we may not find the separating hyperplane between $\mathcal{P}_1$ and $\mathcal{P}_2$, it is possible that a new separating plane is found.

Learning Stackelberg Game Strategies

We present our main algorithm SEPARATE-SEARCH in this section. Briefly speaking, for each action $\mathrm{f}_i \in \mathrm{F}$, $i \in [n]$, we maintain an upper bound $\mathcal{U}_i$ on the feasible region $\mathcal{P}_i$. This upper bound shrinks as we discover a new separating hyperplane between f_i and other effective actions. Meanwhile, we also maintain a lower bound $\mathcal{L}_i$ on $\mathcal{P}_i$, which is the convex hull of all already identified points contained in $\mathcal{P}_i$. More specifically, during the algorithm, we would first figure out the separating hyperplanes between the action we already discovered and we use $\mathcal{U}_i$ to denote the possible feasible region for action $\mathrm{f}_i \in \mathrm{F}$ induced by these separating hyperplanes. Recall that the major difficulty lies in discovering all effective actions, we overcome this by searching through all extreme points in the current upper bound $\mathcal{U}_i$. Thus each time we can either reduce the difference between $\mathcal{U}_i$ and $\mathcal{L}_i$, or discover a new effective action. By doing this, we can get rid of the exponential dependence on L. The formal description of our algorithm SEPARATE-SEARCH is shown in Algorithm 1.

We use $m(S)$ to denote the volume of a set S. During the algorithm, the set $\mathcal{F}$ maintains all the effective actions for the follower which have already been discovered.

To present the sample complexity of our algorithm, we need the following definition.

Definition 3. *For $i \in [n]$, we define $\mathrm{Ext}(A_i)$ to be the number of intersecting points induced by the separating hyperplanes between a_i and $a_{i'}(\forall i' \neq i, i' \in [n])$ and the boundary of $\mathcal{P}$. With slight abuse of notation, we use $\mathrm{Ext}(A_i)$ to denote the set of this points. Furthermore, we define $\mathrm{Ext}(A)$ as*

$$\mathrm{Ext}(A) = \sum_{i=1}^{n} \mathrm{Ext}(A_i).$$

With Theorem 2, we are now ready to prove the main result in this section.

Algorithm 1 SEPARATE-SEARCH

```
1: $\mathcal{U}_i \leftarrow \mathcal{P}, \mathcal{L}_i \leftarrow \emptyset, \forall i \in [n]$.
2:         $\triangleright$ $U_i(\mathcal{L}_i)$ is an upper bound(lower bound) on the feasible region for the follower action $a_i$.
3: Let the leader randomly adopt an strategy $p$ inside $\mathcal{P}$, observe follower response $a_i$ and $\mathcal{F} \leftarrow \{a_i\}$, $\mathcal{L}_i \leftarrow p$.
4: while $\exists \mathcal{U}_i \neq \mathcal{L}_i$ and $\cup_{i\in[n]}\mathcal{L}_i \neq \mathcal{P}$ do
5:     while $\exists i, j \in [n]$ such that $m(\mathcal{U}_i \bigcap \mathcal{U}_j) \neq 0$ and $a_i, a_j \in \mathcal{F}$ do
6:         FIND-SEPARATING-PLANE($\mathcal{U}_i, \mathcal{U}_j, \mathcal{L}_i, \mathcal{L}_j$).
7:     end while
8:     if $\exists \mathcal{U}_i \neq \mathcal{L}_i$ and $i \in \mathcal{F}$ then
9:         EXHAUSTIVE-SEARCH($\mathcal{U}_i, \mathcal{L}_i$).
10:    end if
11: end while
12: return $\mathcal{L}_1, \cdots \mathcal{L}_n$
```

Algorithm 2 EXHAUSTIVE-SEARCH

```
Require: $\mathcal{U}_i, \mathcal{L}_i$
1: for every extreme point $p^\star$ of $\mathcal{U}_i$ and $p^\star \notin \mathcal{L}_i$ do
2:     Let the leader adopt $p^\star$ and observe $\mathrm{f} = \mathrm{f}^*(p^\star)$.
3:     if $\mathrm{f} = a_i$ then
4:         $\mathcal{L}_i \leftarrow$ CONVEXHULL($\mathcal{L}_i, p^\star$).
5:     else if $\mathrm{f} = a_j (j \neq i)$ then
6:         $\mathcal{L}_j \leftarrow$ CONVEXHULL($\mathcal{L}_j, p^\star$).
7:         $\mathcal{F} \leftarrow \mathcal{F} \cup \{a_j\}$.
8:         return
9:     end if
10: end for
11: return
```

Theorem 4. *With $O(mn^2L+\mathrm{Ext}(A))$ samples,* SEPARATE-SEARCH *could identify feasible regions for all action in* F *with high probability. Moreover,* $\mathrm{Ext}(A)$ *is bounded by $n\binom{m+n}{m}$. Thus, the sample complexity for* SEPARATE-SEARCH *is no worse than $O(m^2nL+n\binom{m+n}{m})$.*

We first make the following claim, which will be useful in proving the above theorem.

Claim 5. *During the algorithm, for all $i \in [n]$, $\mathcal{U}_i$, $\mathcal{L}_i$ is convex. Moreover, we have $\mathcal{L}_i \subseteq \mathcal{P}_i \subseteq \mathcal{U}_i$.*

Proof. $\mathcal{U}_i$ is convex since $\mathcal{U}_i$ is the intersection of the half-planes induced by separating hyperplanes and boundary of $\mathcal{P}$. $\mathcal{L}_i$ is convex because $\mathcal{L}_i$ is formed as the convex hull of known points in $\mathcal{P}_i$. Initially, $\mathcal{L}_i$ is empty and $\mathcal{U}_i$ is $\mathcal{P}$. Notice that $\mathcal{U}_i$ shrinks only when a new separating hyperplane is found, and we know $\mathcal{P}_i$ is contained in the same side (with $\mathcal{U}_i$) of this hyperplane. Thus we always have $\mathcal{P}_i \subseteq \mathcal{U}_i$. Meanwhile, notice that all the extreme points of $\mathcal{L}_i$ are contained in $\mathcal{P}_i$, and we know that $\mathcal{P}_i$ is also convex, thus we can conclude that $\mathcal{L}_i \subseteq \mathcal{P}_i$. □

Proof of Theorem 4. According to Claim 5, we know that $\mathcal{U}_i$ ($\mathcal{L}_i$) is indeed an upper (lower) bound on $\mathcal{P}_i$. Another easy observation is that during the execution of SEPARATE-SEARCH, we always maintain $\cup_{i\in[n]}\mathcal{U}_i = \mathcal{P}$ since the union $\cup_{i\in[n]}\mathcal{U}_i = \mathcal{P}$ would not be changed by FIND-SEPARATING-PLANE. Consequently, at the end of our algorithm, we must have $\cup_{i\in[n]}\mathcal{L}_i = \mathcal{P}$, since $\mathcal{U}_i = \mathcal{L}_i$ for all $i \in [n]$ would also implies $\cup_{i\in[n]}\mathcal{L}_i = \mathcal{P}$. As a consequence, by Claim 5 and the fact that $\cup_{i\in[n]}\mathcal{P}_i = \mathcal{P}$, we conclude that when SEPARATE-SEARCH stops, it identifies feasible regions for all $a_i \in \mathcal{F}$. Furthermore, SEPARATE-SEARCH must stop since in each round, we either discover a new effective action, or we ensure that $\mathcal{L}_i = \mathcal{U}_i$ for a new action a_i.

For the sample complexity, each time we invoke FIND-SEPARATING-PLANE, we discover either a new effective action, or a new separating hyperplane. By Theorem 2, the number of samples required is at most $(n^2+n)O(mL) = O(n^2mL)$. The main cost of the SEPARATE-SEARCH comes from the EXHAUSTIVE-SEARCH . For $i \in [n]$, we only sample the extreme points of $\mathcal{U}_i$ and $\mathcal{U}_i$ is induced by separating hyperplanes between a_i and other action in F, as well as the boundary hyperplanes of $\mathcal{P}$. This number is bounded by $\mathrm{Ext}(A_i)$. Moreover, for each $\mathcal{U}_i$, we sample its extreme points once. Thus the total number of samples comes from EXHAUSTIVE-SEARCH is bounded by $O(\mathrm{Ext}(A))$. Moreover, for each $i \in [n]$, there are $n-1$ possible separating hyperplanes from other actions and $m+1$ hyperplanes from $\mathcal{P}$ ($\mathcal{P}$ is a simplex), thus the number of extreme points $\mathrm{Ext}(A_i)$ can be at most $\binom{m+n}{m}$, which further indicates that $\mathrm{Ext}(A)$ can be at most $n\binom{m+n}{m}$, when $m = \Theta(n)$, $\mathrm{Ext}(A)$ is $O(\exp(n))$. □

A direct corollary is that when m or n is constant, i.e., the number of action for the leader or the follower is constant, SEPARATE-SEARCH is efficient, i.e., the required number of sample is $\mathrm{poly}(m, n, L)$.

Corollary 6. *When the number of action for the leader or the follower is constant, the sample complexity for* SEPARATE-SEARCH *is* $\mathrm{poly}(m, n, L)$.

Hardness results

In this section, we provide some negative results about learning the optimal strategy in the Stackelberg game. More specifically, we prove that there is no efficient algorithm to compute the optimal commitment for the leader, i.e., we need to draw exponentially many samples in the worst case. These hardness results indicate that the results in the previous section is in fact the best we can do.

Roughly speaking, the difficulty mainly lies in discovering the new effective action and the fact that we need to fully identify the feasible region for all actions in the worst case. More specifically, consider the following situation: we have not yet discovered any feasible point for an action $a_i \in$ F and we also have not fully identified the feasible region $\mathcal{P}_{i'}$ for another action $a_{i'} \in$ F, but we have an upper bound on $\mathcal{P}_{i'}$, say $\mathcal{U}_{i'}$. Then we may need to check every extreme point of $\mathcal{U}_{i'}$ before we can conclude either $\mathcal{P}_{i'} = \mathcal{U}_{i'}$ or $\mathcal{P}_i$ does not intersect with $\mathcal{U}_{i'}$. Intuitively, the reason is that the feasible region might "hide" in the corner of any extreme point of $\mathcal{U}_{i'}$. Furthermore, these possible feasible regions are disjoint thus the only thing we could do is to exhaustively search through all the extreme points. Finally, in order to prove an exponential lower bound, we need to show that the number of extreme points is exponential in m and n.

Formally, we have the following result.

Theorem 7. *For any algorithm, there exists an instance such that the algorithm must take at least $2^{\Omega(m)}$ samples to compute the optimal strategy to commit to.*

Let's first prove the following lemma before going to the proof of Theorem 7.

Lemma 8. *For any algorithm, there exists an instance such that the algorithm must take at least $2^{\Omega(m)}$ samples to determine whether a particular follower action $a^\star$ is effective.*

Proof. Consider the following Stackelberg game, where the leader has m actions and the follower has $m+2$ actions. The utility function of the follower is

$$U^{\mathrm{f}}(a_i) = (\underbrace{-\tfrac{2}{m-2}, \cdots, -\tfrac{2}{m-2}}_{i-1}, 1, \underbrace{-\tfrac{2}{m-2}, \cdots, -\tfrac{2}{m-2}}_{n-i}), i \le m,$$

$$U^{\mathrm{f}}(a_{m+1}) = \left(-\frac{1}{N^3}, -\frac{1}{N^3}, \cdots, -\frac{1}{N^3}\right),$$

$$U^{\mathrm{f}}(a^\star) = \frac{1}{N^2}X_S,$$

where $S \subseteq [m]$ with $|S| = m/2$ or 0, and X_S is a vector of length m whose i-th element is 1 if $i \in S$ and $-N$ otherwise.

Assume that $N \ge m^3$ is an arbitrarily large constant and all the parameters are known to the leader except S.

Let R_S be the set of leader's commitments such that $a^\star$ is a best response, i.e., $R_S = \{x : a^\star \text{ is a best response to } x\}$.

- If $S = \emptyset$, $a^\star$ is not effective hence $R_S = \emptyset$;
- If $S \ne \emptyset$, the feasible region $R_S \ne \emptyset$.

Then any $(x_1, \cdots, x_m) \in R_S$ must satisfy

$$\forall i \in [m], x_i + \sum_{j\ne i}\left(-\frac{2}{m-2}\right)x_j \le \max\{U^{\mathrm{f}}(a^\star)\} = \frac{1}{N^2}$$

since the utility of choosing $a_i, \forall i \in [m]$ cannot exceed that of choosing $a^\star$. Thus

$$\begin{aligned}\frac{1}{N^2} \ge x_i + \sum_{j\ne i}\left(-\frac{2}{m-2}\right)x_j &= \frac{m}{m-2}x_i - \frac{2}{m-2}\sum_j x_j \\ &= \frac{m}{m-2}x_i - \frac{2}{m-2} \\ \Longrightarrow x_i \le \frac{m-2}{mN^2} + \frac{2}{m} &< \frac{1}{N^2} + \frac{2}{m}. \qquad (*)\end{aligned}$$

Similarly, for action a_{m+1}, we have

$$-\frac{1}{N^3}\sum_{i\in[m]} x_i \le \frac{1}{N^2}\sum_{i\in S} x_i - \frac{1}{N}\sum_{i\notin S} x_i,$$

which yields

$$\sum_{i\in S} x_i \ge 1 - \frac{1}{N} - \frac{1}{N^2}. \qquad (**)$$

Note that for any S with $|S| = m/2$, by condition $(*)$ and $(**)$,

$$\forall i \in S,\ x_i > \frac{2}{m} - \frac{m+2}{2N} - \frac{1}{N^2},$$
$$\forall i \notin S, x_i \leq \frac{1}{N} + \frac{1}{N^2}.$$

In particular, for $N > m^3$ $(m > 1)$, $2/m - (m+2)/2N - 1/N^2 > 1/N + 1/N^2$. Therefore, for any $S_1 \neq S_2$, $R_{S_1} \cap R_{S_2} = \emptyset$.

In other words, to distinguish the cases with S being empty or non-empty, one must sample at least one $x \in R_{S'}$ for all possible non-empty S'. Therefore for any deterministic distinguisher, $\binom{m}{m/2} = 2^{\Omega(m)}$ samples are needed.

For randomized distinguishers, we apply Yao's minimax principle (Yao 1977) and assign the following distribution to the inputs

$$p(S) = \begin{cases} \frac{1}{2} & S = \emptyset \\ \frac{1}{2\binom{m}{m/2}} & |S| = m/2 \end{cases}.$$

With this randomized input, we can check that for any randomized algorithm, the sample complexity is at least $2^{\Omega(m)}$. □

Now, we prove Theorem 7.

Proof of Theorem 7. Consider the following utility matrix for the leader:

$$\begin{array}{ccccc} a_1 & a_2 & \cdots & a_{m+1} & a^\star \\ 1 & 1 & \cdots & 1 & k \\ 1 & 1 & \cdots & 1 & k \\ \vdots & \vdots & \cdots & \vdots & \\ 1 & 1 & \cdots & 1 & k \end{array}$$

where k is a sufficiently large constant.

Then for any algorithm that computes the optimal (or even approximately optimal) leader strategy, it can also answer whether $a^\star$ is effective or not by determining whether the leader's utility is sufficiently larger than 1. Then such an algorithm must take at least $2^{\Omega(m)}$ samples according to Lemma 8.

□

Notice that the R_S in the above proof is in fact a small region around the extreme points of feasible region of a_{m+1}, as pointed out in the above construction, we need to check all of the extreme points in order to make sure the feasibility of $a^\star$, which matches the upper bound of Theorem 4.

Security games

The main inefficiency of SEPARATE-SEARCH comes from EXHAUSTIVE-SEARCH , which plays an exhaustive search for all extreme points. However, as pointed out by Theorem 7, this inefficiency is inevitable and intrinsic to the problem. Nevertheless, we could make the problem tractable by two kinds of relaxations. In this section we explore this possibility and derive efficient sampling schemes. The first kind of relaxation is to make some structural assumption on the game we are playing, while the second is to assume that we have access to other source of data, e.g. noisy estimation of the follower's utility matrix. We discuss the first relaxation in this section and we defer the second relaxation to the appendix. Furthermore, we demonstrate that SEPARATE-SEARCH can serve as a general framework to solve sampling (learning) problems in Stackelberg games, by making some modifications to EXHAUSTIVE-SEARCH , we can possibly make SEPARATE-SEARCH efficient.

Analysis

In this section, we consider the security game and develop an efficient algorithm for the security game. The sample complexity we obtain is $O(n^3 L)$, a significant improvement over $\tilde{O}(n^{6.5} L)$ developed by Blum, Haghtalab, and Procaccia (2014). Moreover, our algorithm is much simpler than theirs, which involves a costly sub-procedure of optimizing a linear function over an unknown convex region, given the initial feasible point and the membership oracle.

Before we proceed, we make the following observation about the difference between security games and general Stackelberg games. In security games we no longer use the polytope of all pure strategies as the initial feasible region, since the dimension could become exponentially large. Instead, we consider $\mathcal{P}$ as the polytope of all feasible coverage vector. $\mathcal{P}$ is convex and it is in a space of dimension n. However, for now, the feasible region $\mathcal{P}$ is no longer a simple simplex, i.e., it can have more than $n + 1$ facets.

Nevertheless, the next thing we are going to show is that replacing EXHAUSTIVE-SEARCH with SECURITY-SEARCH (shown in Algorithm 3) can make SEPARATE-SEARCH efficient in terms of sample complexity. We need to point out that SECURITY-SEARCH is computationally inefficient. However, this inefficiency is inevitable for security games. Korzhyk, Conitzer, and Parr (2010) show that it is NP-hard to compute the optimal strategy even when schedules have size 2. The key insight for Algorithm 3 is that we do not need to search over the entire polytope $\mathcal{U}_i$, instead, one sample is enough. The reason is that we know directly which extreme point could be in the feasible region of another action, if there is one. The formal description for SECURITY-SEARCH is in Algorithm 3.

Now we have the following theorem.

Theorem 9. *For security games with n targets and representation length of L, w.h.p., the modified* SEPARATE-SEARCH *requires $O(n^3 L)$ samples to identify feasible regions for all actions for the attacker.*

Similar to Claim 5, $\mathcal{U}_i$ $(\mathcal{L}_i)$ is still convex and serves as an upper (lower bound) on $\mathcal{P}_i$. the following Claim 10 and Claim 11 serve as the key elements for the proof.

Claim 10. *The strategy $p^\star$ is contained in $\mathcal{U}_i$.*

Proof. Intuitively, $p^\star$ is implementable since the schedule is subset close, which means we can mask an arbitrary coordinate j of p by replacing all schedule D containing j with $D\backslash\{j\}$. Formally, for any pure strategy q, define $C(q)$ to be the set of targets covered by q, it is easy to see that $C(q)$ is

Algorithm 3 SECURITY-SEARCH

```
Require: $\mathcal{U}_i, \mathcal{L}_i$
1: Solve the following linear program and denote the solution by $p$.
       $\max p_i \quad \text{s.t.} \quad p \in \mathcal{U}_i$    (1)
2: $\forall j \in [n]$, set $p^\star_j \leftarrow p_j$ if $j \in \mathcal{F}$, and $p^\star_j \leftarrow 0$ otherwise.
3: Let the defender adopt $p^\star$ and observe $\mathrm{f} = \mathrm{f}^\star(p^\star)$.
4: if $\mathrm{f} = a_i$ then
5:     $\mathcal{L}_i \leftarrow \mathcal{U}_i$.
6: else
7:     Suppose $\mathrm{f} = a_{i'}$, then $\mathcal{L}_{i'} \leftarrow p^\star$ and $\mathcal{F} \leftarrow \mathcal{F} \cup \{a_{i'}\}$.
8: end if
9: return
```

subset close, i.e., for any $T \subseteq C(q)$, we have another pure strategy q' such that $C(q') = T$. Now suppose $p = Ms$, and for $q \in Q$, we set

$$s^\star_q = \begin{cases} 0 & \text{if } C(q) \bigcap (F \backslash \mathcal{F}) \neq \emptyset \\ \sum\limits_{C(\tilde{q}) = C(q)+T,\, T \subseteq F\backslash\mathcal{F}} s_{\tilde{q}} & \text{otherwise} \end{cases}$$

First of all, $s^\star$ is a valid strategy, since

$$\sum_{q \in Q} s^\star_q = \sum_{\substack{q \in Q \\ C(q) \subseteq \mathcal{F}}} \sum_{\substack{C(\tilde{q}) = C(q)+T \\ T \subseteq F\backslash\mathcal{F}}} s_{\tilde{q}} = \sum_{q \in Q} s_q = 1.$$

Then, it is easy to verify that for $i \in F \backslash \mathcal{F}$,

$$p^\star_i = 0$$

and for $i \in \mathcal{F}$,

$$\begin{aligned} p^\star_i &= \textstyle\sum_{q \in Q} M_{iq} \cdot s^\star_q \\ &= \sum_{q \in Q, C(q) \bigcap (F\backslash\mathcal{F}) = \emptyset} M_{iq} \cdot s^\star_q \\ &= \sum_{q \in Q, C(q) \bigcap (F\backslash\mathcal{F}) = \emptyset} M_{iq} \left(\sum_{C(\tilde{q}) = C(q)+T, T \subseteq F\backslash\mathcal{F}} s_{\tilde{q}} \right) \\ &= \textstyle\sum_{q \in Q} M_{iq} \cdot s_q \\ &= p_i. \end{aligned}$$

Therefore, $p^\star$ is implementable. Moreover, according to Algorithm 1, when SECURITY-SEARCH is called, all $\mathcal{U}_i$'s do not intersect with one another, and $\mathcal{U}_i$ contains all defender strategies that the attacker choosing a_i is the best response in $\mathcal{F}$. Given defender strategy $p^\star$, the attacker would prefer to attack the target i to any other target $i' \in \mathcal{F}$, because the utility of attacking any target i depends only on p_i and we know $p^\star_i = p_i$, which implies $p \in \mathcal{U}_i$. □

Claim 11. *If* $\mathrm{f} = a_i$, *then* $\mathcal{L}_i = \mathcal{U}_i$.

Proof. For any $j \in F \backslash \mathcal{F}$, we claim that

$$p^\star = \arg\min_{p \in \mathcal{U}_i} (U^a(p,i) - U^a(p,j)). \tag{2}$$

As a consequence, for any $p \in \mathcal{U}_i$ and any $j \in F \backslash \mathcal{F}$,

$$U^a(p,i) - U^a(p,j) \geq U^a(p^\star,i) - U^a(p^\star,j) \geq 0$$

where the second inequality comes from $\mathrm{f} = a_i$. For any $j \in \mathcal{F}$,

$$U^a(p,i) - U^a(p,j) \geq 0$$

since we have already separated i and j. Combining the above two equations, we know that $\forall p \in \mathcal{U}_i$ choosing a_i is always the best response for the attacker, with which we can conclude that $\mathcal{U}_i = \mathcal{L}_i$.

To prove (2), notice that for $j \in F \backslash \mathcal{F}$, $p^\star_j = 0$, and

$$\begin{aligned} U^a(p,j) &= p_j U^a_c(j) + (1-p_j) U^a_u(j) \\ &= U^a_u(j) + (U^a_u(j) - U^a_c(j)) p_j, \end{aligned}$$

thus $U^a(p,j)$ is decreasing with p_j. Consequently $p^\star$ already maximizes $U^a(p,j)$ in $\mathcal{U}_i$. On the other hand, since $p^\star_i = p_i$, we have $U^a(p,i) = U^a(p^\star_i, i)$. Furthermore, we choose p according to (1) and it minimizes $U^a(p,i)$ over $\mathcal{U}_i$, thus $p^\star$ also minimize $U^a(p,i)$. So

$$U^a(p,i) \geq U^a(p^\star,i) \geq U^a(p^\star,j) \geq U^a(p,j),$$

where the second equation holds because $\mathrm{f} = a_i$. □

Proof of Theorem 9. Combining Claim 10 and Claim 11 would directly imply the correctness of the modified SEPARATE-SEARCH.

Finally, consider the sample complexity for the modified SEPARATE-SEARCH, since each call of SECURITY-SEARCH only requires one sample and it would either discover a new effective action of the follower or fully identify the feasible region of a discovered action, thus we make at most $2n$ calls to SECURITY-SEARCH . Consequently, the worst case complexity is dominated by FIND-SEPERATING-PLANE, which is $O(n^3 L)$. □

Conclusion

In this paper, we propose an algorithm for finding the optimal strategy to commit to for the leader in Stackelberg games. Although our algorithm still has exponential dependence on m in the worst case, it makes exponential improvement over previous results and gets rid of the exponential dependence on L. Moreover, it is guaranteed to be efficient in certain cases. Furthermore, we also give an impossibility result showing that the inefficiency (the exponential dependence on m) is intrinsic to the problem itself. Our algorithm is quite general, when applied to security games, which have richer structures than general Stackelberg games, it gives a better sample complexity than previous results.

One interesting future research direction is to further apply our algorithm SEPARATE-SEARCH to other related problems that dealing with uncertainties in Stackelberg (security) games. In the appendix, we point out a possible extension, which utilizes other source of data as well, and derive some preliminary results.

References

Amin, K.; Singh, S.; and Wellman, M. P. 2016. Gradient methods for stackelberg security games. In *UAI*, 2–11.

An, B.; Pita, J.; Shieh, E.; Tambe, M.; Kiekintveld, C.; and Marecki, J. 2011. Guards and protect: Next generation applications of security games. *ACM SIGecom Exchanges* 10(1):31–34.

Balcan, M.-F.; Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2015. Commitment without regrets: Online learning in stackelberg security games. In *ACM EC*, 61–78. ACM.

Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2014. Learning optimal commitment to overcome insecurity. In *NIPS*, 1826–1834.

Blum, A.; Haghtalab, N.; and Procaccia, A. D. 2015. Learning to play stackelberg security games.

Camerer, C. F.; Ho, T.-H.; and Chong, J.-K. 2004. A cognitive hierarchy model of games. *The Quarterly Journal of Economics* 119(3):861–898.

Chen, L.; Lin, F.; Tang, P.; Wang, K.; Wang, R.; and Wang, S. 2017. K-memory strategies in repeated games. In *AAMAS*, 1493–1498.

Chen, X.; Deng, X.; and Teng, S.-H. 2009. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)* 56(3):14.

Chen, L.; Tang, P.; and Wang, R. 2017. Bounded rationality of restricted turing machines. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, 444–450.

Conitzer, V., and Sandholm, T. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*, 82–90. ACM.

Conitzer, V., and Sandholm, T. 2008. New complexity results about nash equilibria. *Games and Economic Behavior* 63(2):621–641.

Daskalakis, C.; Goldberg, P. W.; and Papadimitriou, C. H. 2009. The complexity of computing a nash equilibrium. *SIAM Journal on Computing* 39(1):195–259.

Goodfellow, I.; Bengio, Y.; Courville, A.; and Bengio, Y. 2016. *Deep learning*, volume 1. MIT press Cambridge.

Haghtalab, N.; Fang, F.; Nguyen, T. H.; Sinha, A.; Procaccia, A. D.; and Tambe, M. 2016. Three strategies to success: Learning adversary models in security games. In *IJCAI*, volume 16, 308–314.

Jain, M.; Tsai, J.; Pita, J.; Kiekintveld, C.; Rathi, S.; Tambe, M.; and Ordónez, F. 2010. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40(4):267–290.

Jiang, A. X., and Leyton-Brown, K. 2011. Polynomial-time computation of exact correlated equilibrium in compact games. In *Proceedings of the 12th ACM conference on Electronic commerce*, 119–126. ACM.

Kalai, A. T., and Vempala, S. 2006. Simulated annealing for convex optimization. *Mathematics of Operations Research* 31(2):253–266.

Kamra, N.; Gupta, U.; Fang, F.; Liu, Y.; and Tambe, M. 2018. Policy learning for continuous space security games using neural networks. In *AAAI-18*.

Korzhyk, D.; Conitzer, V.; and Parr, R. 2010. Complexity of computing optimal stackelberg strategies in security resource allocation games. In *AAAI*, volume 10, 805–810.

Letchford, J., and Conitzer, V. 2010. Computing optimal strategies to commit to in extensive-form games. In *Proceedings of the 11th ACM conference on Electronic commerce*, 83–92. ACM.

Letchford, J.; Conitzer, V.; and Munagala, K. 2009. Learning and approximating the optimal strategy to commit to. In *International Symposium on Algorithmic Game Theory*, 250–262. Springer.

Ling, C. K.; Fang, F.; and Kolter, J. Z. 2018. What game are we playing? end-to-end learning in normal and extensive form games. In *IJCAI-18*, 396–402.

Marecki, J.; Tesauro, G.; and Segal, R. 2012. Playing repeated stackelberg games with unknown opponents. In *IJCAI-12*, 821–828.

Papadimitriou, C. H., and Roughgarden, T. 2008. Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)* 55(3):14.

Pita, J.; Jain, M.; Marecki, J.; Ordóñez, F.; Portway, C.; Tambe, M.; Western, C.; Paruchuri, P.; and Kraus, S. 2008. Deployed armor protection: the application of a game theoretic model for security at the los angeles international airport. In *IJCAI-08*, 125–132.

Shen, W.; Tang, P.; and Zuo, S. 2018. Computer-aided mechanism design: designing revenue-optimal mechanisms via neural networks. *CoRR* abs/1805.03382.

Sinha, A.; Kar, D.; and Tambe, M. 2016. Learning adversary behavior in security games: A pac model perspective. In *AAMAS*, 214–222.

Tambe, M. 2011. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press.

Vershynin, R. 2018. *High-dimensional probability: An introduction with applications in data science*, volume 47. Cambridge University Press.

Xu, H.; Rabinovich, Z.; Dughmi, S.; and Tambe, M. 2015. Exploring information asymmetry in two-stage security games. In *AAAI*, 1057–1063.

Yang, R.; Ford, B.; Tambe, M.; and Lemieux, A. 2014. Adaptive resource allocation for wildlife protection against illegal poachers. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, 453–460. International Foundation for Autonomous Agents and Multiagent Systems.

Yao, A. C.-C. 1977. Probabilistic computations: Toward a unified measure of complexity. In *Foundations of Computer Science, 1977., 18th Annual Symposium on*, 222–227. IEEE.

Yin, Y.; Xu, H.; Gan, J.; An, B.; and Jiang, A. X. 2015. Computing optimal mixed strategies for security games with dynamic payoffs. In *IJCAI*, 681–688.

Zheng, S.; Waggoner, B.; Liu, Y.; and Chen, Y. 2017. Active information acquisition for linear optimization. *arXiv preprint arXiv:1709.10061.*

Zuo, S., and Tang, P. 2015. Optimal machine strategies to commit to in two-person repeated games. In *AAAI*, 1071–1078.

Appendix

Bayesian Cases

In this section, we extend the results about the Stackelberg game to the Bayesian case, where the follower can have multiple types and the type of the follower is drawn from an underlying distribution. We use $\Theta(|\Theta| = \tau)$ to denote the set of types. Follow the work in (Letchford, Conitzer, and Munagala 2009), we consider two possible extensions. The results derived in this section is a simple combination of the results developed in (Letchford, Conitzer, and Munagala 2009) and 4, but for completeness, we present the proof here.

First, we consider the following simplified version of the Bayesian case, in which each sample would contain the best reponse for any type of the follower. In other words, we can get the best response sample for the follower of any type. Assuming owning the access to such powerful samples, we would have the following results.

Theorem 12. *Using $O(\tau(\mathrm{Ext}(A) + mn^2L))$ samples, we could learn all the required separating planes for all types of the follower. With these separating planes, we could compute the optimal strategy for the leader.*

Proof. This is a simple application of SEPARATE-SEARCH. For the follower type $\theta^1, \cdots, \theta^\tau$, we run SEPARATE-SEARCH for each type and by 4, we can learn the separating planes for each type by $O(\mathrm{Ext}(A) + mn^2L)$ samples and $O(\tau(\mathrm{Ext}(A) + mn^2L))$ are required in total. □

The sample complexity in (Letchford, Conitzer, and Munagala 2009) is $O(\tau(V^{-1}n \log n + mn^2L))$, thus the 12 gets an exponential improvement in terms of L in the worst case.

Second, we consider the more general case, in which each sample contains the best response sample for the follower with type θ, where θ is drawn from some known distribution $P(\theta)$. Define

$$\tilde{P} = \max_{\theta \in \Theta} P^{-1}(\theta).$$

Now we have the following theorem.

Theorem 13. *With probability $1 - \delta$, we could learn all the required separating planes for all types of the follower with $O(\tilde{P}\tau(\mathrm{Ext}(A)+mn^2L) \log(\tau(\mathrm{Ext}(A)+mn^2L)\delta^{-1}))$ samples,*

Proof. We adopt the same strategy as in 12. The main difference here is that one sample might not be enough to ensure that we get a best response sample for any type $\theta \in \Theta$. Instead, we draw $\tilde{P} \log(\tau(\mathrm{Ext}(A)+mn^2L)\delta^{-1})$ samples each time and the probability that we do not get a best response sample for aby desired type θ' is bounded by

$$\begin{aligned} &(1 - P(\theta'))^{\tilde{P} \log(\tau(\mathrm{Ext}(A)+mn^2L)\delta^{-1})} \\ \le &(1 - P(\theta'))^{P^{-1}(\theta') \log(\tau(\mathrm{Ext}(A)+mn^2L)\delta^{-1})} \\ \le &e^{-\log(\tau(\mathrm{Ext}(A)+mn^2L)\delta^{-1})} \\ = &\frac{\delta}{\tau(\mathrm{Ext}(A) + mn^2L)} \end{aligned}$$

By the union bound, the total failure probability is bounded by

$$\frac{\delta}{\tau(\mathrm{Ext}(A) + mn^2L)} \cdot \tau(\mathrm{Ext}(A) + mn^2L) = \delta$$

□

Again, there is an exponential improvement in terms of L against the results in (Letchford, Conitzer, and Munagala 2009).

Subgaussian Sample

In this section, we further seek another applications of SEPARATE-SEARCH. Instead of knowing the exact structure of the game we are playing(like the Stackelberg security game), we assume that we have access to some noisy samples of follower's utility matrix U^{f}, i.e. we can sample each entry of U^{f}, and get an estimation of U^{f}_{ij}. We call this sample an estimation sample and we assume that it is drawn from a σ-subgaussian, with means equals to U^{f}_{ij}.[3] In the rest of this section, we assume $\sigma = 1$ for simplicity and it is easy to extend the results to more general case.

Of course, with the sample access to the entry of the follower utility matrix U^{f}, we can directly estimate each entry of U^{f}. However, the naive approach would require us to draw as much as 2^L samples, in order to get a good estimation to precision L, which could be highly inefficient. Moreover, since there are lots of uncertainties if we only use estimation samples, it is still not well understood how to efficiently utilize the estimation samples only in the literature. Nevertheless, if we combine these two sources of samples together, we can possible make the sampling procedure efficient. Again, we would utilize SEPARATE-SEARCH as the general procedure and make some heuristics to EXHAUSTIVE-SEARCH . Before proceeding to our algorithm, we first introduce some known techniques that our algorithm relies on.

Known Results

We need the result from (Zheng et al. 2017). In their paper, they consider the following partially-specified optimization problem.

$$\max_x c^T x, \text{s.t.} Ax \le b, x \ge 0 \qquad (3)$$

[3]Distribution D with mean μ is σ subgaussian, if have $\mathrm{E}[e^{t(X-\mu)}] \le e^{\sigma^2 t^2/2}$ for all t. The family of subgaussian distribution captures many natural arise distribution, like all bounded distribution which are supported on $[0, \sigma]$ and Gaussian distributions with variance σ^2, see (Vershynin 2018) for more about subgaussian.

Both A and b is known in advance, but c is unknown. Nevertheless, we have access to samples of each entry of c and the sample draws from an σ-subgaussian distribution with mean equal to the true value of c_i. Formally, they have

Theorem 14. *(restated, (Zheng et al. 2017)) With probability $1-\delta$, the algorithm* ALALO *is able to find the optimal solution to 3, and would draws at most the following number of samples*

$$LOW(A)\log\Delta^{-1}\left(\log|S^{(1)}|+\log\delta^{-1}+\log\log\Delta^{-1}\right).$$

where $S^{(1)}$ is the set of all extreme points of the feasible region and Δ is the gap in objective value between the optimal extreme point and the second-best,

$$x^\star = arg\max_{x\in S^{(1)}} c^Tx, \quad \Delta = c^Tx^\star - \max_{x\in S^{(1)}\setminus x^\star} c^Tx.$$

and LOW(A) is the solution for the following convex programming

$$\begin{aligned}\min_{\tau} \quad & \sum_{i=1}^n \tau_i \\ s.t. \quad & \sum_{i=1}^n \frac{(x_i-y_i)}{\tau_i} \le \left(c^T(x-y)\right)^2, \forall x,y\in S^{(1)} \\ & \tau_i \ge 0\,\forall i\in[n].\end{aligned}$$

Our results

Back to our results, we would replace EXHAUSTIVE-SEARCH with SUBGAUSSIAN-SEARCH , which is formally described in 4.

Algorithm 4 SUBGAUSSIAN-SEARCH

```
Require: $\mathcal{U}_i, \mathcal{L}_i$
 1: for $j \in F\backslash\mathcal{F}$ do
 2:     $p_j = \text{ALALO}(\mathcal{U}_i, U^{\text{f}}_{\cdot j} - U^{\text{f}}_{\cdot i}, \delta/2n^2)$   ▷ $U^{\text{f}}_{\cdot j}$ denote the $j$-th column of $U^{\text{f}}$
 3:     Sample $p_j$
 4:     if $a_{p^\star} = a_{i'} \neq a_i$ then
 5:         $\mathcal{L}_{i'} \leftarrow p_j$ and $\mathcal{F} \leftarrow \mathcal{F} \cup \{a_{i'}\}$
 6:         return
 7:     end if
 8: end for
 9: $\mathcal{L}_i = \mathcal{U}_i$
10: return
```

We make the following definition here.

Definition 15. *Define Low(A_j) to be the solution for the following convex programming.*

$$\begin{aligned}\min_{\tau} \quad & \sum_{i=1}^n \tau_i \\ s.t. \quad & \sum_{i=1}^n \frac{(x_i-y_i)}{\tau_i} \le \left(c^T(x-y)\right)^2, \forall x,y\in \text{Ext}(A_j) \qquad (4)\\ & \tau_i \ge 0\,\forall i\in[n].\end{aligned}$$

Furthermore, we define

$$Low(A) = \max_{j\in[n]} Low(A_j)$$

Now, we have the following theorem.

Theorem 16. *With probability $1-\delta$,* SEPARATE-SEARCH *identifies feasible regions for all actions in* F. *It requires $O(mn^2L)$ best response samples and the number of eastimation sample is*

$$\tilde{O}\left((m+n)n^2L\cdot Low(A)\log\delta^{-1}\right)$$

Proof. First of all, we show that the invocation of ALALO is valid. Since subgaussian random variables are additive, each entry of the objective is $\sqrt{2}$-subgaussian. We could use two samples of U^{f}_{ki} and U^{f}_{kj} to simulate a sample for $c_k = U^{\text{f}}_{kj} - U^{\text{f}}_{ki}$.

For the correctness of the algorithm, from 14 we know that with probability $1-\delta/2n^2$,

$$p_j = \arg\max_{p\in U_i} U^{\text{f}}_{\cdot j}x - U^{\text{f}}_{\cdot i}x.$$

Thus, if $a_{p_j} = a_i$, with probability $1-\delta/2n^2$, a_j is not feasible in U_i. Consequently, if all the best response samples we get is a_i, then with probability $1-\delta/2n$, $U_i = L_i = \mathcal{P}_i$. Moreover, since we invoke SUBGAUSSIAN-SEARCH for at most $2n$ times, the algorithm is correct with probability $1-\delta$ by union bounds.

Now, we turn to the sample complexity. For best response samples, each invocation of SUBGAUSSIAN-SEARCH consumes at most n best response samples. As a result, the number of sample required is dominant by FIND-SEPERATING-PLANE and $O(mn^2L)$ samples are required. Consider the number of estimation samples, there are at most $2n^2$ evocation of ALALO. By 14, each invocation requires the following number of samples

$$\text{Low}(U_i)\log\Delta^{-1}(\log|S^{(1)}|+\log\delta^{-1}+\log n+\log\log\Delta^{-1})$$

By definition, Low(A) is greated than Low(U_i) for any possible U_i during the algorithm, since the extreme points of U_i are contained in Ext(A_i), which indicates that 4 is always an upper bound on Low(U_i). The $\log\Delta^{-1}$ term is bounded by the representation length L. Moreover, we know that

$$\log|S^{(1)}| \le \log\binom{m+n}{m} \le (m+n)\log n.$$

since U_i is induced by at most $m+1$ boundary hyperplanes and $n-1$ separating planes. Putting things together, the number of estimation sample required is

$$O(n^2L\cdot\text{Low(A)}\left((m+n)\log m + \log n + \log L + \log\delta^{-1}\right)$$

For abbreviation, it is

$$\tilde{O}\left((m+n)n^2L\cdot\text{Low(A)}\log\delta^{-1}\right)$$

□

A Simple Lower Bound for Security Games

In this section, we prove a simple lower bound on the sample complexity for security games, formally, we have

Theorem 17. *Any algorithm requires $\tilde{\Omega}(nL)$ samples to identify the optimal commitment in Stackelberg security games.*

Proof. Consider the following instance. The defender is allowed to defend at most one target among n targets. Moreover, the defender's utility is given as follow:

$$U_c^d(t) = U_u^d(t) = \begin{cases} 10 & t = 1 \\ 1 & t \in \{2, 3, \cdots, n\} \end{cases}$$

Let $x = (x_1, \cdots, x_n)$ denotes the optimal strategy for the defender, it is easy to see that $x_1 = 0$ and $|x| = 1$. Consider the following family of attacker's utility:

$$U_c^a(1) = U_u^a(1) = 0$$

$$U_c^a(t) = -1, U_u^a(t) = \frac{y_t}{1 - y_t}, \forall t \in \{2, 3, \cdots, n\}$$

where y_t satiesfies $2^L y_t \in \mathbb{N}^+$ and $\sum_{i=1}^n y_i = 1$.

For any feasible $\{y_t\}$, the optimal strategy for the defender would be $(0, y_2, \cdots, y_n)$ and the defender's utility would be 10 under this strategy. Any other strategy would be suboptimal, since attacker would attack the first target only if for any $t \in \{2, \cdots, n\}$

$$\begin{aligned} U^a(t) &= x_t U_c^a(t) + (1 - x_t) U_u^a(t) \\ &= -x_t + (1 - x_t) \frac{y_t}{1 - y_t} \\ &= \frac{1}{1 - y_t}(y_t - x_t) \\ &\leq U^a(1) = 0 \end{aligned}$$

i.e., $x_t \geq y_t$ for any $t \in \{2, \cdots, n\}$. The number of possible configurations for y_t is $N = \binom{2^L - 1}{n-2} \approx 2^{nL}/n!$. For any deterministic sampling algorithm, we can view the sample process as a decision tree, the degree for each node is bounded by n(since there are at most n different outcomes). Consequently, the depth of the decision tree would be $\Omega(\log_n N) \approx \tilde{\Omega}(nL)$. The similar things work for any randomize algorithm, we only need to apply Yao's Minmax principal(Yao 1977) and assign the uniform distribution for all possible configurations. In conclusion, any algorithm needs $\tilde{\Omega}(nL)$ samples to identify the optimal commitment in Stackelberg security games. $\square$