Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-Form Games

Jiří Čermák[1], **Branislav Bošanský**[1,2], **Karel Durkota**[1], **Viliam Lisý**[1,3], **Christopher Kiekintveld**[4]
[1] Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague
[2] Department of Computer Science, Aarhus University
[3] Department of Computing Science, University of Alberta
[4] Department of Computer Science, University of Texas at El Paso
jiri.cermak@agents.fel.cvut.cz, branislav.bosansky@agents.fel.cvut.cz, karel.durkota@agents.fel.cvut.cz,
lisy@ualberta.ca, cdkiekintveld@utep.edu

Abstract

Strong Stackelberg Equilibrium (SSE) is a fundamental solution concept in game theory in which one player commits to a strategy, while the other player observes this commitment and plays a best response. We present a new algorithm for computing SSE for two-player extensive-form general-sum games with imperfect information (EFGs) where computing SSE is an NP-hard problem. Our algorithm is based on a correlated version of SSE, known as Stackelberg Extensive-Form Correlated Equilibrium (SEFCE). Our contribution is therefore twofold: (1) we give the first linear program for computing SEFCE in EFGs without chance, (2) we repeatedly solve and modify this linear program in a systematic search until we arrive to SSE. Our new algorithm outperforms the best previous algorithms by several orders of magnitude.

Introduction

The roles of players in many games are often asymmetric. One example is the ability of one player (*the leader*) to commit to a strategy, to which the other players (*the followers*) react by playing their best response. In real-world scenarios, the leader can model a market leader with the power to set the price for items or services, or a defense agency committing to a security protocol to protect critical facilities. Optimal strategies for the players in such situations are described by the Strong Stackelberg Equilibrium (SSE) (Leitmann 1978; von Stengel and Zamir 2004). There are many examples of successful applications of SSE in infrastructure protection (Tambe 2011) as well as protecting the environment and wildlife (Fang, Stone, and Tambe 2015).

In most of the existing works, the game models are simplified and do not consider the sequential interaction among players (Pita et al. 2008; Tsai et al. 2009; Shieh et al. 2012; Jiang et al. 2013). One reason is computational complexity, since computing SSE is often NP-hard when sequential interaction is allowed (Letchford and Conitzer 2010). Another reason is the lack of practical algorithms. The only algorithm designed specifically for computing SSE in two-player imperfect-information general-sum extensive-form games (EFGs) was only introduced recently (Bosansky and Cermak 2015) and formulates the problem as a mixed-integer variant of sequence-form linear program (referred to as BC15). However, the scalability of BC15 is limited as it contains a binary variable for each sequence of actions of the follower.

Our main contribution is a novel algorithm computing SSE in EFGs that offers a dramatic speed-up in computation time compared to BC15. The key idea behind our algorithm is in computing a variant of SSE, where the leader commits to correlated strategies – i.e., the leader can send signals to the follower, and the best response of the follower is to follow these signals. We use this variant to find the original SSE by systematically restricting which signals the leader can send to the follower. This variant of SSE has previously been studied in single step games (Conitzer and Korzhyk 2011), finite turn-based and concurrent-move games (Bosansky et al. 2015), infinite concurrent-move stochastic games (Letchford et al. 2012), and security games (Xu et al. 2015; Rabinovich et al. 2015; Durkota et al. 2015). Formally, it has been defined as Stackelberg Extensive-Form Correlated Equilibrium (SEFCE) in (Bosansky et al. 2015). While it was shown that the utility value for the leader in SSE cannot be closely approximated by SEFCE (Letchford et al. 2012), we show that one can use SEFCE to compute SSE. Since there was no previously known algorithm for computing SEFCE in EFGs, we also show that SEFCE can be found in polynomial time in EFGs without chance.

The paper is structured as follows. After introducing the formalism of EFGs, we formally define both SSE and SEFCE, and give an example of a game where these concepts differ. Next, we present the linear program (LP) for computing SEFCE in EFGs without chance (we describe a modified LP for EFGs with chance in the full version of the paper). Afterwards, we present our algorithm for computing SSE in EFGs that iteratively solves the LP for SEFCE with additional constraints until SSE is reached. Finally, we provide three variants of our algorithm and show that each variant significantly improves the computation time compared to BC15 on randomly generated games and an example of a search game.

Technical Background

Extensive-form games model sequential interactions between players and can be visually represented as game trees. Formally, a two-player EFG is defined as a tuple $G = (\mathcal{N}, \mathcal{H}, \mathcal{Z}, \mathcal{A}, u, \mathcal{C}, \mathcal{I})$: $\mathcal{N} = \{l, f\}$ is a set of players, the leader and the follower. We use i to refer to one of the players, and $-i$ to refer to his opponent. $\mathcal{H}$ denotes a finite set of *nodes* in the game tree. Each node corresponds to a unique *history* of actions taken by all players and chance from the root of the game; hence, we use the terms history and node interchangeably. We say that h is a *prefix* of h' ($h \sqsubseteq h'$) if h lies on a path from the root of the game tree to h'. $\mathcal{A}$ denotes the set of all actions. $\mathcal{Z} \subseteq \mathcal{H}$ is the set of all *terminal nodes* of the game. For each $z \in \mathcal{Z}$ we define a *utility function* for each player i ($u_i : \mathcal{Z} \to \mathbb{R}$). Chance player selects actions based on a fixed probability distribution known to all players. Function $\mathcal{C} : \mathcal{H} \to [0, 1]$ denotes the probability of reaching node h due to chance; $\mathcal{C}(h)$ is the product of chance probabilities of all actions in history h.

Imperfect observation of player i is modeled via *information sets* $\mathcal{I}_i$ that form a partition over $h \in \mathcal{H}$ where i takes action. Player i cannot distinguish between nodes in any information set $I \in \mathcal{I}_i$. We overload the notation and use $A(I_i)$ to denote possible actions available in each node from information set I_i. We assume that action a uniquely identifies the information set where it is available. We assume *perfect recall*, which means that players remember history of their own actions and all information gained during the course of the game. As a consequence, all nodes in any information set I_i have the same history of actions for player i.

Pure strategies Π_i assign one action for each $I \in \mathcal{I}_i$. A more efficient representation in the form of *reduced pure strategies* Π_i^* assigns one action for each $I \in \mathcal{I}_i$ reachable while playing according to this strategy. A *mixed strategy* $\delta_i \in \Delta_i$ is a probability distribution over Π_i. For any pair of strategies $\delta \in \Delta = (\Delta_l, \Delta_f)$ we use $u_i(\delta) = u_i(\delta_i, \delta_{-i})$ for the expected outcome of the game for player i when players follow strategies δ. A *best response* of player i to the opponent's strategy δ_{-i} is a strategy $\delta_i^{BR} \in BR_i(\delta_{-i})$, where $u_i(\delta_i^{BR}, \delta_{-i}) \geq u_i(\delta_i', \delta_{-i})$ for all $\delta_i' \in \Delta_i$.

Strategies in EFGs with perfect recall can be compactly represented by using the sequence form (Koller, Megiddo, and von Stengel 1996). A *sequence* $\sigma_i \in \Sigma_i$ is an ordered list of actions taken by a single player i in history h. $\emptyset$ stands for the empty sequence (i.e., a sequence with no actions). A sequence $\sigma_i \in \Sigma_i$ can be extended by a single valid action a taken by player i, written as $\sigma_i a = \sigma_i'$. We say that σ_i is a *prefix* of σ_i' ($\sigma_i \sqsubseteq \sigma_i'$) if σ_i' is obtained by finite number (possibly zero) of extensions of σ_i. We use $\sigma_i(I_i)$ and $\sigma_i(h)$ to denote the sequence of i leading to I_i and h, respectively. We use the function $I_i(\sigma_i')$ to obtain the information set in which the last action of the sequence σ_i' is taken. For an empty sequence, function $I_i(\emptyset)$ returns the information set of the root node. A mixed strategy of a player can now be represented as a *realization plan* ($r_i : \Sigma_i \to \mathbb{R}$). A realization plan for a sequence σ_i is the probability that player i will play σ_i under the assumption that the opponent plays to allow the actions specified in σ_i to be played. By $g_i : \Sigma_l \times \Sigma_f \to \mathbb{R}$ we denote the *extended utility function*, $g_i(\sigma_l, \sigma_f) = \sum_{z \in \mathcal{Z} | \sigma_l(z) = \sigma_l \wedge \sigma_f(z) = \sigma_f} u_i(z) C(z)$. If no leaf is reachable with pair of sequences σ, value of g_i is 0.

Solution Concepts in EFGs

Here we provide a formal definition of Strong Stackleberg Equilibrium (SSE) (e.g., in (Leitmann 1978)) and Stackelberg Extensive-Form Correlated Equilibrium (SEFCE) (Bosansky et al. 2015) and give the intuition on an example game.

Definition 1. *A strategy profile $\delta = (\delta_l, \delta_f)$ is a* Strong Stackelberg Equilibrium *if δ_l is an optimal strategy of the leader given that the follower best-responds. Formally:*

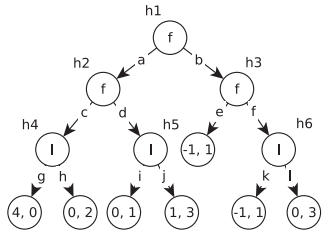
$$(\delta_l, \delta_f) = \underset{\delta_l' \in \Delta_l, \delta_f' \in BR_i(\delta_l')}{\arg\max} u_l(\delta_l', \delta_f'). \quad (1)$$

The SSE of the game in Figure 1 (the first utility in every leaf is for the leader, second for the follower) prescribes the leader to commit to playing g in h_4, j in h_5, and k in h_6. The strategy of the follower is then to play a in h_1 and d in h_2, leading to the expected utility of 1 for the leader.

In SEFCE we allow the leader to send signals to the follower and condition his strategy on sent signals. More specifically, the leader chooses $\pi_f^* \in \Pi_f^*$ as the recommendations for the follower according to SEFCE before the game starts. The actual recommendation to play some action $a \in A(I_f)$ is revealed to the follower only after he reaches I_f. Therefore, the follower only knows the past and current recommendations, and the probability distribution from which the recommendations are drawn in the future.

Definition 2. *A probability distribution λ on reduced pure strategy profiles Π^* is called a* Stackelberg Extensive-Form Correlated Equilibrium *if it maximizes the leader's utility subject to the constraint that whenever play reaches an information set I where the follower can act, the follower is recommended an action a according to λ such that the follower cannot gain by unilaterally deviating from a in I and possibly in all succeeding information sets given the posterior on the probability distribution of the strategy of the leader, defined by the actions taken by the leader so far.*

The middle table in Figure 1 represents the distribution λ forming the SEFCE of the example game (rows are labeled by Π_l^*, columns by Π_f^*). The leader chooses the signals to the follower to be either $\{a, c\}$ or $\{a, d\}$ based on the probability distribution depicted in the table. Afterwards, the corresponding column defines a valid mixed strategy for the leader and the signals the follower receives in his information sets. More specifically, the follower can receive either c or d in h_2. When the follower receives the recommendation to play c, the leader commits to mix uniformly between g and h in h_4 and to play i in h_5. When the follower receives d as the recommendation, the leader commits to playing g in h_4 and j in h_5. Finally in h_1 the follower is recommended to play a, while the leader commits to play k in h_6 to ensure that the follower does not deviate from playing a. The expected utility of the leader is 1.5 in SEFCE. Note that SSE in this representation always corresponds to a table where only a single column of the follower has non-zero values.



	$\{a,c\}$	$\{a,d\}$	$\{b,e\}$	$\{b,f\}$
$\{g,i,k\}$	0.25	0	0	0
$\{g,i,l\}$	0	0	0	0
$\{g,j,k\}$	0	0.5	0	0
$\{g,j,l\}$	0	0	0	0
$\{h,i,k\}$	0.25	0	0	0
$\{h,i,l\}$	0	0	0	0
$\{h,j,k\}$	0	0	0	0
$\{h,j,l\}$	0	0	0	0

	$\emptyset$	(a)	(b)	(a,c)	(a,d)	(b,e)	(b,f)
$\emptyset$	1	1	0	0.5	0.5	0	0
(g)	0.75	0.75	0	0.25	0.5	-	-
(h)	0.25	0.25	0	0.25	0	-	-
(i)	0.5	0.5	0	0.5	0	-	-
(j)	0.5	0.5	0	0	0.5	-	-
(k)	1	1	0	-	-	0	0
(l)	0	0	0	-	-	0	0

Figure 1: (Left) EFG with different SEFCE and SSE. (Middle) SEFCE distribution over Π^*. (Right) SEFCE correlation plan.

LP for Computing SEFCE

To compactly represent the behavior described in the middle table in Figure 1, we use a *correlation plan* (von Stengel and Forges 2008) that is quadratic in the size of the game tree, instead of the exponential representation using Π^*. A correlation plan for a sequence pair $p(\sigma_l, \sigma_f)$ represents the expected probability that σ_l will be played if actions from σ_f are recommended to the follower. In order to model SEFCE strategies using the correlation plan, the follower must be able to determine whether following the signal is the best response. Therefore, we need to specify the plan for so called *relevant sequences*. Consider our game in Figure 1; when the follower receives, for example, signal c in h_2, the follower needs to know the commitment of the leader in the related part of the tree – i.e, in both nodes h_4, h_5 – to evaluate whether following the signal is the best response.

Definition 3. *A pair of sequences (σ_l, σ_f) is termed* relevant *if and only if either $\sigma_f = \emptyset$, or $\exists h, h' \in \mathcal{H}, h' \sqsubseteq h; \sigma_l = \sigma_l(h) \wedge h' \in I_f(\sigma_f)$.*

By $rel(\sigma_i)$ we denote the set of sequences of $-i$ which form a relevant pair with σ_i. In our example $rel((a)) = rel((b)) = \Sigma_l$, $rel((b,e)) = rel((b,f)) = \{\emptyset, (k), (l)\}$, and $rel((a,c)) = rel((a,d)) = \{\emptyset, (g), (h), (i), (j)\}$.

Definition 4. *A* correlation plan *(von Stengel and Forges 2008) is a partial function $p : \Sigma_l \times \Sigma_f \to \mathbb{R}$ such that there is a probability distribution λ on the set of reduced strategy profiles Π^* so that for each relevant sequence pair (σ_l, σ_f), the term $p(\sigma_l, \sigma_f)$ is defined and fulfills $p(\sigma_l, \sigma_f) = \sum_{(\pi_l, \pi_f) \in \Pi^*} \lambda(\pi_l, \pi_f)$ where π_l, π_f prescribe playing all of the actions in σ_l and σ_f, respectively.*

Let us now describe the SEFCE strategies (middle table in Figure 1) using the correlation plan (the right table; rows are labeled by Σ_l, columns by Σ_f). Every column of the table corresponds to an expected probability of the occurrence of the leader's sequences when the follower follows his recommendations in the future, and gets the recommendation to play the σ_f corresponding to this column. We use '-' to mark irrelevant sequence pairs. The entry for every σ_l, σ_f is the sum of all the entries corresponding to the pure strategies containing all the actions from σ_l and σ_f in the middle table of Figure 1. The behavior in columns corresponding to sequences (a,c) and (a,d) matches the behavior discussed in the previous section. The behavior in columns for sequence $\emptyset$ and (a) corresponds to the expected probability of playing sequences of the leader given the probability of recommendations, e.g., the probability of playing g for the recommendation (a) is equal to $p((g),(a,c)) + p((g),(a,d))$ (in this case the leader will play uniformly either g with probability 0.5 according to the column for (a,c) or g with probability 1 according to the column for (a,d)). Probabilities in column for (a) allow the follower to evaluate his choices in h_1.

Now we are ready to describe the LP for computing SEFCE in EFGs without chance that uses correlation plan:

$$\max_{p,v} \sum_{\sigma_l \in \Sigma_l} \sum_{\sigma_f \in \Sigma_f} p(\sigma_l, \sigma_f) g_l(\sigma_l, \sigma_f) \tag{2}$$

$$\text{s.t.} \quad p(\emptyset, \emptyset) = 1; \quad 0 \le p(\sigma_l, \sigma_f) \le 1 \tag{3}$$

$$p(\sigma_l(I), \sigma_f) = \sum_{a \in A(I)} p(\sigma_l(I)a, \sigma_f) \;\; \forall I \in \mathcal{I}_l, \forall \sigma_f \in rel(\sigma_l) \tag{4}$$

$$p(\sigma_l, \sigma_f(I)) = \sum_{a \in A(I)} p(\sigma_l, \sigma_f(I)a) \;\; \forall I \in \mathcal{I}_f, \forall \sigma_l \in rel(\sigma_f) \tag{5}$$

$$v(\sigma_f) = \sum_{\sigma_l \in rel(\sigma_f)} p(\sigma_l, \sigma_f) g_f(\sigma_l, \sigma_f) + \sum_{I \in \mathcal{I}_f;\, \sigma_f(I) = \sigma_f} \sum_{a \in A_f(I)} v(\sigma_f a) \;\; \forall \sigma_f \in \Sigma_f \tag{6}$$

$$v(I, \sigma_f) \ge \sum_{\sigma_l \in rel(\sigma_f)} p(\sigma_l, \sigma_f) g_f(\sigma_l, \sigma_f(I)a) + \sum_{I' \in \mathcal{I}_f;\, \sigma_f(I') = \sigma_f(I)a} v(I', \sigma_f)$$
$$\forall I \in \mathcal{I}_f, \forall \sigma_f \in \bigcup\nolimits_{h \in I} rel(\sigma_l(h)), \forall a \in A(I) \tag{7}$$

$$v(\sigma_f(I)a) = v(I, \sigma_f(I)a) \qquad \forall I \in \mathcal{I}_f, \forall a \in A(I) \tag{8}$$

The LP is derived from the computation of Extensive-Form Correlated Equilibrium (von Stengel and Forges 2008) by omitting the incentive constraints for the leader and maximizing the expected utility of the leader (this is similar to the approach in normal-form games (Conitzer and Korzhyk 2011)). Constraints (3) to (5) ensure that p is a well-formed correlation plan. Constraint (6) ensures that $v(\sigma_f)$ represents the expected utility of playing σ_f for the follower, when he follows his recommendations. The first sum represents the expected utility of the leaves reached by playing according to σ_l and σ_f, the second sum represents the contribution of the expected utility from information sets reached by the continuations of σ_f. Constraint (7) ensures that $v(I_f, \sigma_f)$ is the maximum over all possible sequences leaving I_f (denoted as $\sigma_f(I_f)a$ for all $a \in A(I_f)$) after the follower has received recommendation σ_f. Finally, constraint (8) forces the move recommended to the follower in I_f to be optimal.

Definition 5. *We say that p uses* inconsistent recommendation *in $I \in \mathcal{I}_f$ if and only if p defines two different recommendations for the follower in I. Formally, $\exists a, a' \in A(I), a \neq a', \exists \sigma_l, \sigma'_l \in \bigcup_{h \in I} \sigma_l(h)\ p(\sigma_l, \sigma_f(I)a) > 0 \wedge p(\sigma'_l, \sigma_f(I)a') > 0$. If there exists no such information set we say that p uses only* consistent recommendations.

Theorem 1. *Assume a solution of the LP as described in eqs.* (2) *to* (8) *such that there are only consistent recommendations for the follower. There is a SSE strategy that can be found in polynomial time from the p variables.*

Proof. First, we show how the strategy of the leader is constructed. In every $I \in \mathcal{I}_l$ there is a subset Σ_r of relevant sequences $rel(\sigma_l(I))$ played with a positive probability. The behavior in I is specified by $p(\sigma_l(I)a, \sigma_f)$ for all $a \in A(I)$ for arbitrary $\sigma_f \in \Sigma_r$, as the behavior is the same for all $\sigma'_f \in \Sigma_r$. This is guaranteed by constraint (5) – it forces the probability of $p(\sigma_l, \sigma'_f)$ to be equal to the sum of $p(\sigma_l, \sigma''_f)$ over all extensions σ''_f of σ'_f. Since there can be only a single extension played with positive probability (assumed consistent recommendations) the probabilities must be equal.

For every $I \in \mathcal{I}_f$ there exists at most one action $a \in A(I)$ with $p(\sigma_l, \sigma_f a) > 0$ for some $\sigma_l \in \Sigma_l$ and $\sigma_f = \sigma_f(I)$ (consistent recommendations). By taking these actions and arbitrary actions in information sets where there is no such action a, we obtain a pure strategy for the follower. Finally, due to the correctness of the strategy of the leader proved in the previous step and constraints (6–8), this pure strategy is a best response of the follower. □

Theorem 2. *Assume a solution of the LP as described in eqs.* (2) *to* (8)*. The objective value is greater than or equal to the expected utility of the leader in SSE.*

Proof. Theorem 1 shows that in case the leader uses only consistent recommendations, the value of the LP corresponds to the expected utility of the leader in SSE. If the leader can also use inconsistent recommendations, the value of the LP can be only greater or equal. □

Algorithm Computing SSE

In this section we describe the algorithm for computing SSE. The algorithm uses the linear program for computing SEFCE in case the game is without chance. Otherwise, a slightly modified version of this LP is required, however, the algorithm remains the same. Due to the space constraints, we describe this modified LP in details in the full version of the paper and refer to one of these two LPs as UB-SSE-LP.

The high level idea of our algorithm (depicted in Algorithm 1) is a recursive application of the following steps: (1) solve the UB-SSE-LP, (2) detect the set of information sets of the follower with inconsistent recommendations $\mathcal{I}_{in}$, (3) restrict the leader to use only consistent recommendations in $\mathcal{I}_{in}$ by adding new constraints to the UB-SSE-LP. Restrictions are added cumulatively, until we arrive at a restricted UB-SSE-LP yielding only consistent p. The expected utility for the leader and the correlation plan p in this solution correspond to a candidate for the expected utility of the leader and the strategies of players in SSE. It is only a solution candidate, since we have enforced actions, which may not be a part of SSE.

Input: An UB-SSE-LP P
Output: leader's expected utility and strategy profile in SSE

```
1  M ← {(∞, ∅)}; LB ← −∞; p_c ← ∅
2  while M ≠ ∅ do
3      (UB, m) ← max(M)
4      if UB < LB then
5          return (LB, p_c)
6      apply(m, P)
7      if feasible(P) then
8          (value, p) ← solve(P)
9          I_in ← inconsistentRecommendations(p)
10         if I_in = ∅ then
11             if value > LB then LB ← value; p_c ← p
12         else addRestrictions((UB, m), M, I_in, value)
13     revert(m, P)
14 return (LB, p_c)
```

Algorithm 1: Algorithm for computing the SSE.

In more details, Algorithm 1 assumes as the input the UB-SSE-LP P for the game we want to solve. By *modification* $mod = (UB, m)$ we denote a pair of constraints m, to be added to P, and the upper bound UB on the value of P after adding the constraints. M is the set of modifications to be explored during the search for the SSE sorted in descending order of UB. The variable LB stores the expected utility for the leader in the best solution candidate found so far p_c. The main cycle of the algorithm starts on line 2, we iterate until there are no possible modifications of P left. On line 3 we remove the modification with the highest UB from M. We choose such $mod = (UB, m)$ in order to first explore modifications with the potential to lead to solution candidates with the highest expected utility for the leader. The algorithm verifies whether the modification mod (the one with the highest UB value) can improve the current best solution candidate (line 4). If not, the algorithm terminates and the best candidate found so far is the SSE. Otherwise, we add the constraints in m to P (line 6). If the modified P is feasible, the algorithm solves the LP (line 8) obtaining the expected utility of the leader and the correlation plan p. We find the set of information sets where the follower gets an inconsistent recommendation in p (line 9). If p uses only consistent recommendations, this solution corresponds to a solution candidate. If the expected utility for the leader is higher than for the best solution candidate found so far, we replace it (line 11). If $\mathcal{I}_{in}$ is not empty, we generate new modifications to be applied to P and add them to M (line 12). The function $addRestrictions$ will be discussed in more detail in the next subsection, as we explored several options of the modification generation. Finally, we revert the changes in m (line 13). If there are modifications left to be explored in M we continue with the next iteration. Every modification contains all the constraints added to the original P given as an input, therefore after $revert$ we again obtain the original P.

When we enforce the whole strategy of the follower to be consistent with the SSE, the solution of the LP corresponds to the SSE, as it now maximizes the expected utility

```
1  addRestrictions((UB, m), M, I_in, value)
2      I ← getShallowest(I_in)
3      for a ∈ A(I) do
4          UB_a ← value; m_a ← m
5          for σ_l ∈ rel(σ_f(I)a) do
6              m_a ← m_a ∪ {p(σ_l, σ_f(I)) = p(σ_l, σ_f(I)a)}
7          M ← M ∪ {(UB_a, m_a)}
```

Algorithm 2: SI-LP.

of the leader under the restriction the follower gets recommended the pure best response to the strategy of the leader. It remains to be shown, that we are guaranteed to add modifications to M which force the correct set of actions of the follower in every version of $addRestrictions$. The final correctness arguments will be provided after discussing the $addRestrictions$ versions used.

Rules for Restricting Follower's Behavior

We examine different approaches for method $addRestrictions$ that generates new modifications of UB-SSE-LP resulting in three different variants of our new algorithm.

In Algorithm 2 we describe the first version of the function $addRestrictions$, labeled SI-LP. On line 2 we find the shallowest information set I, where the follower receives an inconsistent recommendation. We generate modification for every $a \in A(I)$. Every such modification enforces corresponding $a \in A(I)$ to be recommended deterministically. The upper bound is set to the value of P computed before invoking $addRestrictions$. The shallowest information set is chosen to avoid unnecessary modifications in deeper parts of the game tree, which might end up not being visited at all. This version of $addRestrictions$ transforms Algorithm 1 to branch and bound algorithm.

By adding $mod = (UB, m)$ to M for every action in the shallowest information set with inconsistent recommendations, until no such set exists, we ensure that all of the actions which might form a part of SSE will be tried. The behavior in information sets with consistent recommendation need not be restricted, as we are sure that the follower has no incentive to deviate and therefore plays his best response maximizing the expected utility of the leader. Finally, since we assign to UB a value which forms an upper bound on the solution of P after adding constraints m, we are sure that if we terminate the algorithm on line 4 in Algorithm 1, there is indeed no possibility to encounter a solution candidate better than the one yielding the current LB.

A second option, presented in Algorithm 3, chooses $\mathcal{I}_c \subseteq \mathcal{I}_{in}$ (line 2) and restricts the recommendations in every $I \in \mathcal{I}_c$. The restriction in I is done in the following way. First, detect the subset A_c of $A(I)$ of actions which are recommended with positive probability (line 5) and make the recommendation mutually exclusive using binary variables (lines 8 and 9), converting the LP P to a mixed integer linear program (MILP). We use two options of creating $\mathcal{I}_c$. First, we create a singleton containing only the shallowest $I \in \mathcal{I}_{in}$, we refer to this algorithm as SI-MILP. Second, we let $\mathcal{I}_c = \mathcal{I}_{in}$, we refer to this algorithm as AI-MILP. Algorithm 1 using both SI-MILP and AI-MILP closely resembles

```
1  addRestrictions((UB, m), M, I_in, value)
2      I_c ← chooseSets(I_in)
3      UB' ← ∞; m' ← m
4      for I ∈ I_c do
5          A_c ← {a ∈ A(I) | ∃σ_l ∈ Σ_l p(σ_l, σ_f(I)a) > 0}
6          for a ∈ A_c do
7              for σ_l ∈ rel(σ_f(I)a) do
8                  m' ← m' ∪ {p(σ_l, σ_f(I)a) ≤ b_a}
9              m' ← m' ∪ {Σ_{a∈A_c} b_a = 1}
10         M ← M ∪ {(UB', m')}
```

Algorithm 3: MILP.

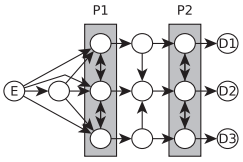
constraint generation, with the difference that additional binary variables are also added in every iteration.

If we introduce a binary variable for every action of the follower in the game, we are guaranteed to obtain the SSE, as the MILP then finds a strategy profile maximizing the expected utility of the leader (ensured by the objective), while the follower plays a pure best response to the strategy of the leader (breaking ties in favor of the leader due to the objective), which is the definition of the SSE. If we create some partial enforcement of consistent recommendations using the binary variables and we obtain a pure strategy for the follower then this is again SSE, since the enforcement in the rest of the game would not make any difference as the follower already gets consistent recommendations there. Finally, since we restrict the follower's recommendations until consistent recommendations are obtained, both MILP based rules indeed guarantee to find the SSE.

Experimental Evaluation

We now turn to the experimental evaluation of the three described variants of our algorithm for computing an SSE. We use BC15 (Bosansky and Cermak 2015) as a baseline, state-of-the-art algorithm for computing SSE in EFGs. Single-threaded IBM CPLEX 12.5 solver was used to compute all the (MI)LPs. We use two different domains previously used for evaluation of BC15: a search game representing the scenario where security units defend several targets against an attacker, and randomly generated games.

Search Game. The search game is played on a directed graph (see Figure 2). The follower aims to reach one of the destination nodes (D1 – D3) from starting node (E) in a given number of moves, while the leader aims to catch the follower with one of the two units operating in the shaded areas of the graph (P1 and P2). The follower receives different reward for reaching different destination node (the reward is randomly selected from the interval $[1, 2]$). The leader receives positive reward 1 for capturing the follower. Once the follower runs out of moves without reaching any goal or being captured, both players receive 0. The follower leaves tracks in the visited nodes that can be discovered if the leader visits the node. The follower can erase the tracks in the current node (it takes one turn of the game). The follower does not know the position of the patrolling units, the leader observes only the tracks left by the follower.



Algs \ Steps	4	5	6
BC15	25	863	42,261
SI-LP	9	212	10,790
AI-MILP	6	257	11,997
SI-MILP	5	272	11,263

Figure 2: (Left) Search game graph. (Right) Runtimes in seconds for the search game with increasing depth.

Table 1: Number of games solved in given time intervals.

Algs\Runtime	1s	5s	30s	2min	25min	4h
BC15	2	12	137	245	393	55
SI-LP	583	191	54	12	4	0
AI-MILP	529	259	51	5	0	0
SI-MILP	483	279	72	9	1	0

Randomly Generated Games. We use randomly generated games, where in each state of the game the number of available actions is randomly generated up to a given parameter $\{2, \ldots, \max_A\}$. Each action leads to a state where the opponent is to move and also generates *an observation* for the opponent. An observation is a number from a set $\{1, \ldots, \max_O\}$ and determines partitioning of the nodes into the information sets – for player i, the nodes h with the same history of moves $\sigma_i(h)$ and the observations generated by the actions of the opponent $-i$ belong to the same information set. We generate games of differing sizes by varying parameters $\max_A = \{3, 4\}$, $\max_O = \{2, 3\}$, and depth of the game (up to 5 actions for each player). The utility for the players is randomly generated in the interval $[-100, 100]$. The utilities are correlated with factor set to -0.5 (1 represents identical utilities, -1 zero-sum utilities).

Results

The runtime results on random games are depicted in the top graph of Figure 3. The x-axis shows the number of realization plans of the follower, while the y-axis depicts the time in seconds needed to solve a given instance (both axes are logarithmic). The number of realization plans of the follower is a good estimate of how difficult the game is to solve as it reflects both the size of the game as well as the structure of information sets. Each point represents the mean time needed for every algorithm to solve the instances from a given time interval (at least 600 different instances). The standard errors of the mean values are very small compared to the differences between algorithms and not visible in the graph.

The results show that each of the variants significantly outperforms the previous state-of-the-art algorithm BC15. It typically takes around 10 minutes for BC15 to solve games with 10^6 realization plans of the follower, while our algorithms were often able to find solutions under a second. AI-MILP performs best on average, since it is the least sensitive to the different structure of the instances solved. AI-MILP fixes the behavior in a higher number of information sets and so it preforms a successful trade off between the complexity of solving a single MILP and the number of MILP invocations. The second best approach on average is the SI-MILP.

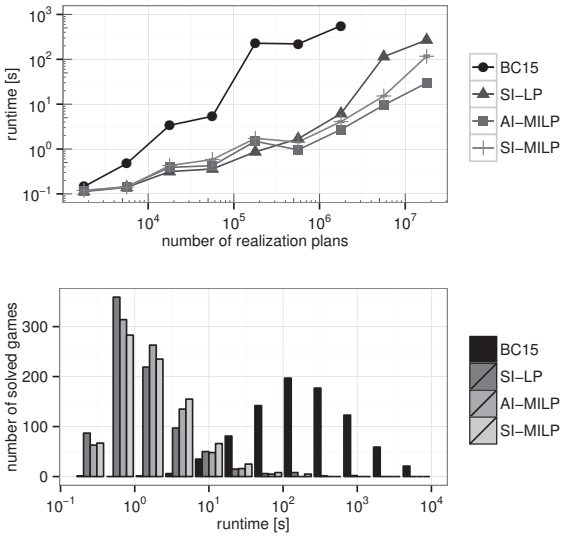


Figure 3: (Top) Runtimes on randomly generated games. (Bottom) Number of solved games in given time intervals.

The average performance is slightly worse due to a higher number of MILP invocations needed to solve more difficult instances (i.e., the ones with many inconsistent recommendations). Finally, the SI-LP has the worst average performance out of the variants of our new algorithm, since SI-LP needs even more LP invocations on more difficult instances in comparison to the previous variants of our algorithm.

Additionally, we provide in the bottom graph of Figure 3 a histogram for number of instances solved (y-axis) within a time interval (x-axis). The results were calculated on random games with the number of realization plans from interval $[3 \cdot 10^5, 3 \cdot 10^6]$. Despite a slightly worse averge performance of SI-LP on these instances, it solved the highest number of instances very fast compared to the other two variants (SI-LP solves 69% of the instances under 1 second, while AI-MILP solves 62% and SI-MILP 57%). The histogram also shows the reason behind the worse average performance of SI-LP. There are multiple instances that SI-LP solves in more then 200 seconds, while such outliers are not present for the latter two variants (the worst outlier for SI-LP took 773 seconds, while the longest time for SI-MILP was 146 seconds, for AI-MILP 57 seconds and for BC15 2.5 hours). For clarity we provide the same data in coarser intervals in Table 1, where the outliers are clearly visible (the label of column represents the upper bound of the corresponding time interval, the label of the column to the left the lower bound of the time interval). The results show that SI-LP is more efficient on instances where the advantage in using the correlated strategies is marginal and there are only few information sets with inconsistent recommendations. On the other hand, AI-MILP offers higher robustness across different instances.

Finally, in Figure 2 we present the results on the search game. All the new approaches performed similarly, outperforming the BC15 in every setting. This shows that our al-

gorithm outperforms BC15 even in an unfavorable setting. The search game has a specific structure, where the strategy space of the leader is large (joint actions of the patrolling units), while the strategy space of the follower is significantly smaller. This structure is favorable for BC15, since it implies a relatively small number of binary variables (the MILP contains one binary variables for each sequence of the follower), with the overall size of the MILP being linear in the size of the game, while the size of our underlying LP is quadratic due to the correlation plan.

Conclusion

We present a novel domain-independent algorithm for computing Strong Stackelberg Equilibria (SSE) in extensive-form games that uses the correlated variant of Stackelberg Equilibria (Stackelberg Extensive-Form Correlated Equilibrium). This work opens several areas for future research. First, our algorithm can be adapted and applied for solving specific domains since its scalability is significantly better in comparison to the existing algorithms. Second, the scalability can most-likely be further improved by employing iterative approaches for solving the underlying linear program. Third, several question were not addressed in our approach and remain open: Is it possible to generalize presented algorithm for computing SSE with multiple followers? Can we relax the assumption of perfect recall?

Acknowledgments

This research was supported by the Czech Science Foundation (grant no. 15-23235S), by the Danish National Research Foundation and The National Science Foundation of China (under the grant 61361136003) for the Sino-Danish Center for the Theory of Interactive Computation and Office of Naval Research Global (grant no. N62909-13-1-N256). This material is based upon work supported by the National Science Foundation (grant no. IIS-1253950). This research was supported by Alberta Innovates Technology Futures through the Alberta Innovates Centre for Machine Learning and Reinforcement Learning and AI Lab and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS15/205/OHK3/3T/13 and SGS15/206/OHK3/3T/13.

References

Bosansky, B., and Cermak, J. 2015. Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games. In *AAAI Conference on Artificial Intelligence*, 805–811.

Bosansky, B.; Branzei, S.; Hansen, K. A.; Miltersen, P. B.; and Sorensen, T. B. 2015. Computation of Stackelberg Equilibria of Finite Sequential Games. In *11th Conference on Web and Informatics (WINE)*.

Conitzer, V., and Korzhyk, D. 2011. Commitment to Correlated Strategies. In *AAAI Conference on Artificial Intelligence*.

Durkota, K.; Lisy, V.; Bosansky, B.; and Kiekintveld, C. 2015. Approximate solutions for attack graph games with imperfect information. In *Decision and Game Theory for Security*, 228–249. Springer.

Fang, F.; Stone, P.; and Tambe, M. 2015. When Security Games Go Green: Designing Defender Strategies to Prevent Poaching and Illegal Fishing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI*, 2589–2595.

Jiang, A. X.; Yin, Z.; Zhang, C.; Tambe, M.; and Kraus, S. 2013. Game-theoretic Randomization for Security Patrolling with Dynamic Execution Uncertainty. In *12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 207–214.

Koller, D.; Megiddo, N.; and von Stengel, B. 1996. Efficient Computation of Equilibria for Extensive two-person Games. *Games and Economic Behavior* 247–259.

Leitmann, G. 1978. On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications* 26(4):637–643.

Letchford, J., and Conitzer, V. 2010. Computing Optimal Strategies to Commit to in Extensive-Form Games. In *11th ACM conference on Electronic commerce*, 83–92.

Letchford, J.; MacDermed, L.; Conitzer, V.; Parr, R.; and Isbell, C. L. 2012. Computing Optimal Strategies to Commit to in Stochastic Games. In *AAAI Conference on Artificial Intelligence*.

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 *7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 125–132.

Rabinovich, Z.; Jiang, A. X.; Jain, M.; and Xu, H. 2015. Information Disclosure as a Means to Security. In *14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 645–653.

Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; DiRenzo, J.; Maule, B.; and Meyer, G. 2012. Protect: A deployed game theoretic system to protect the ports of the united states. In *11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 13–20.

Tambe, M. 2011. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press.

Tsai, J.; Kiekintveld, C.; Ordóñez, F.; Tamble, M.; and Rathi, S. 2009. IRIS - A Tool for Strategic Security Allocation in Transportation Networks Categories and Subject Descriptors. In *8th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 37–44.

von Stengel, B., and Forges, F. 2008. Extensive-form Correlated Equilibrium: Definition and Computational Complexity. *Mathematics of Operations Research* 33(4):1002–1022.

von Stengel, B., and Zamir, S. 2004. Leadership with Commitment to Mixed Strategies. Technical report, CDAM Research Report LSE-CDAM-2004-01.

Xu, H.; Rabinovich, Z.; Dughmi, S.; and Tambe, M. 2015. Exploring Information Asymmetry in Two-Stage Security Games. In *AAAI Conference on Artificial Intelligence*.