Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games

Branislav Bošanský[1,2], **Jiří Čermák**[1]
[1] Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague
[2] Department of Computer Science, Aarhus University
bosansky@cs.au.dk, jiri.cermak@agents.fel.cvut.cz

Abstract

Stackelberg equilibrium is a solution concept prescribing for a player an optimal strategy to commit to, assuming the opponent knows this commitment and plays the best response. Although this solution concept is a cornerstone of many security applications, the existing works typically do not consider situations where the players can observe and react to the actions of the opponent during the course of the game. We extend the existing algorithmic work to extensive-form games and introduce novel algorithm for computing Stackelberg equilibria that exploits the compact sequence-form representation of strategies. Our algorithm reduces the size of the linear programs from exponential in the baseline approach to linear in the size of the game tree. Experimental evaluation on randomly generated games and a security-inspired search game demonstrates significant improvement in the scalability compared to the baseline approach.

Introduction

Solving games and computing game-theoretic solutions is one of the key topics of artificial intelligence. The best known solution concepts in game theory is Nash equilibrium that prescribes optimal *strategy profile* (one strategy for each player) such that no player can gain anything by unilaterally changing their strategy. However, a line of research that analyzes two-player games where the first player (termed *the leader*) is able to commit to a publicly known strategy before the other player (termed *the follower*) moves has received a significant focus in recent years. This solution concept is known as Stackelberg (or leader-follower) equilibrium (von Stackelberg 1934; Leitmann 1978; von Stengel and Zamir 2004) and it is a cornerstone of many security applications of game theory. The examples include airport security (Pita et al. 2008), assigning Air Marshals to flights (Tsai et al. 2009), or protecting the coast (Shieh et al. 2012).

Most of the existing applications of Stackelberg equilibria (SE) typically focus on single-step normal-form or Bayesian games where the players do not learn anything during the course of the game – the players are not able to observe the actions performed by the opponent and cannot react to this information. Such an assumption can be too strong in practice, since conditioning the strategies of the players by the current development in the game can be desirable (e.g., security procedures based on a suspicious behavior). Game theory provides representation for modeling finite sequential interactions between the players known as *extensive-form games* (EFGs). This model is sufficiently generic to represent uncertainty in observation (i.e., players are not able to perfectly observe the current state of the game), or stochastic environment.

The problem of computing SE in EFGs has been studied from the computational perspective and it was shown that computing SE in extensive-form games with imperfect information and stochastic events is NP-hard (Letchford and Conitzer 2010). However, a tailored algorithm for computing an exact SE in EFGs is currently missing. A baseline approach is thus to transform an EFG to a single step normal-form game and use one of the existing approaches based on computing multiple linear programs (LPs) (Conitzer and Sandholm 2006), or formulating the problem as a mixed-integer linear program (MILP) (Paruchuri et al. 2008). This transformation is, however, exponential and we show that the scalability of the baseline approach is very limited.

This paper aims to address this issue and provides the first specific algorithm for computing SE in two player EFGs. Our approach exploits the compact representation of strategies known as *the sequence form* (Koller, Megiddo, and von Stengel 1996; von Stengel 1996). We generalize two existing algorithms for computing SE in single-step games and introduce two variants of our novel algorithm: (1) based on solving multiple LPs, and (2) based on reformulating the problem as a MILP. Moreover, the size of our MILP is linear in the size of the game, and it contains only binary integer variables; hence, it provides an upper bound on the computational complexity of the problem of computing SE and shows that it is an NP-complete problem (Karp 1971).

We first give necessary technical background on EFGs, Nash and Stackelberg solution concepts, following by the description of the existing algorithms for computing each of these concepts. Next, we describe our novel algorithm that exploits the sequence form for computing SE in EFGs. Finally, we experimentally demonstrate the scalability of our algorithm on randomly generated games and a security-inspired search game. The results show that our MILP algorithm is significantly faster and can solve games several magnitudes larger compared to the baseline approach.

Technical Background

Extensive-form games (EFGs) model sequential interactions between the players and can be visually represented as game trees. Nodes in the game tree represent the states of the game; each state of the game corresponds to a sequence of moves executed by all players in the game. Each node is assigned to a player that acts in the game state associated with this node. An edge from a node corresponds to an action that can be performed by the player who acts in this node. EFGs model limited observations of the players by grouping certain nodes into *information sets*; a player cannot distinguish between nodes that belong to the same information set. The model also represents uncertainty about the environment and stochastic events by using a special *Nature player*.

Formally, a two-player EFG is defined as a tuple $G = (N, H, Z, A, \rho, u, \mathcal{C}, \mathcal{I})$: $N = \{1, 2\}$ is a set of two players; we use i to refer to one of the two players (either 1 or 2), and $-i$ to refer to the opponent of i. 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 Nature from the root of the game; hence, we use the terms history and node interchangeably. We denote by $Z \subseteq H$ the set of all *terminal nodes* of the game. A denotes the set of all actions and we overload the notation and use $A(h) \subseteq A$ to represent the set of actions available to the player acting in node $h \in H$. We specify $ha = h' \in H$ to be node h' reached from node h by performing action $a \in A(h)$. For each terminal node $z \in Z$ we define a *utility function* for each player i ($u_i : Z \to \mathbb{R}$).

The function $\rho : H \to N \cup \{c\}$ assigns each node to a player who takes an action in the node, where c means that the Nature player selects an action in the node based on a fixed probability distribution known to all players. We use function $\mathcal{C} : H \to [0, 1]$ to denote the probability of reaching node h due to Nature (i.e., assuming that both players play all required actions to reach node h). The value of $\mathcal{C}(h)$ is defined to be the product of the probabilities assigned to all actions taken by the Nature player in history h.

Imperfect observation of player i is modeled via *information sets* $\mathcal{I}_i$ that form a partition over the nodes assigned to player i $\{h \in H : \rho(h) = i\}$. Every information set contains at least one node and each node belongs to exactly one information set. Nodes in an information set of a player are not distinguishable to the player. All nodes h in a single information set $I_i \in \mathcal{I}_i$ have the same set of possible actions $A(h)$; hence, an action a from $A(h)$ uniquely identifies information set I_i and there cannot exist any other node $h' \in H$ that does not belong to information set I_i and for which a is allowed to be played (i.e., $a \in A(h')$). We overload the notation and use $A(I_i)$ to denote the set of actions defined for each node h in this information set. We assume *perfect recall*, which means that players perfectly remember 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.

Strategies and Solution Concepts in EFGs

Solving a game implies finding a strategy profile (i.e., one strategy for each player) that satisfies conditions given by a specific solution concept. *Pure strategies* (denoted as Π_i) correspond to assignments of exactly one action for each information set. A mixed strategy is a probability distribution over the set of all pure strategies of a player. We denote by Δ_i the set of all mixed strategies of player i. For any pair of strategies $\delta \in \Delta = (\Delta_1, \Delta_2)$ 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 $\delta_i^{BR} = BR_i(\delta_{-i})$, for which $u_i(\delta_i^{BR}, \delta_{-i}) \geq u_i(\delta_i', \delta_{-i})$ for all strategies $\delta_i' \in \Delta_i$.

Nash equilibrium (NE) is the best known solution concept in game theory and it describes the behavior of agents under certain assumptions about their rationality. In a Nash equilibrium, every player plays a best response to the strategies of the other players. Formally, a strategy profile $\delta = (\delta_1, \delta_2)$ is a NE if and only if for each player i it holds that δ_i is a best response to δ_{-i} (i.e., $\forall i \in N\ \delta_i = BR_i(\delta_{-i})$).

In the *Stackelberg* setting, the first player commits to a certain strategy δ_1 that is observed by the second player that plays a pure best response π_2 to this strategy ($\pi_2 = BR_2(\delta_1)$). Two strategies (δ_1, π_2) are in a *Stackelberg equilibrium* (SE) if $\pi_2 = BR_2(\delta_1)$ and the expected utility of the leader is maximal (i.e., $\forall \delta_1' \in \Delta_1, \forall \pi_2' \in \Pi_2$ such that $\pi_2' = BR_2(\delta_1')$ it holds $u_1(\delta_1, \pi_2) \geq u_1(\delta_1', \pi_2')$). In case player 2 has multiple possible best response strategies, we assume the ties are broken in favor of player 1. This assumption is common in the literature and the equilibrium is called *Strong Stackelberg Equilibrium* (SSE) (Leitmann 1978; Conitzer and Sandholm 2006; Paruchuri et al. 2008; Yin et al. 2010).

Sequence-Form and Computing NE in EFGs

Strategies in EFGs with perfect recall can be compactly represented by using the notion of sequences (Koller, Megiddo, and von Stengel 1996; von Stengel 1996). A *sequence* σ_i is an ordered list of actions taken by a single player i in history h. The number of actions (i.e., the length of sequence σ_i) is denoted as $|\sigma_i|$, the empty sequence (i.e., a sequence with no actions) is denoted as $\emptyset$. The set of all possible sequences for player i is denoted by Σ_i. A sequence $\sigma_i \in \Sigma_i$ can be extended by a single action a taken by player i, denoted by $\sigma_i a = \sigma_i'$. In games with perfect recall, all nodes in an information set I_i share the same sequence of actions for player i and we use $\mathsf{seq}_i(I_i)$ to denote this sequence. We overload the notation and use $\mathsf{seq}_i(h)$ to denote the sequence of actions of player i leading to node h. Since action a uniquely identifies information set I_i and all nodes in an information set share the same history, each sequence uniquely identifies an information set. We use the function $\mathsf{inf}_i(\sigma_i')$ to denote the information set in which the last action of the sequence σ_i' is taken. For an empty sequence, function $\mathsf{inf}_i(\emptyset)$ denotes the information set of the root node. A mixed strategy of a player can now be represented as a probability distribution over the sequences and it is called a *realization plan* (denoted $r_i : \Sigma_i \to \mathbb{R}$). A realization plan for a sequence σ_i is the probability that player i will play this sequence of actions under the assumption that the opponent plays in a way which allows the actions specified in σ_i to be played.

When the strategies are represented as realization plans, we can compute a Nash equilibrium of a two-player general-sum extensive-form game using a linear complimentary program (LCP) of a polynomial size in the size of the game tree using the sequence form (Koller, Megiddo, and von Stengel 1996; von Stengel 1996). To describe the program we need to define payoff function $g_i : \Sigma \rightarrow \mathbb{R}$ that extends the utility function to all nodes in the game tree. The payoff function g_i represents the expected utility of all nodes reachable by sequentially executing the actions specified in a pair of sequences σ:

$$g_i(\sigma_i, \sigma_{-i}) = \sum_{h \in Z \,:\, \forall j \in N \;\; \sigma_j = \mathsf{seq}_j(h)} u_i(h) \cdot \mathcal{C}(h) \qquad (1)$$

The value of the payoff function is defined to be 0 if no leaf is reachable by sequentially executing all of the actions in the sequences σ – i.e., either all actions from the pair of sequences σ are executed and an inner node $h \in H \setminus Z$ is reached, or there is no further action that can be executed from a node that is reached during the execution of the sequences. We say that a pair of sequences σ is *compatible* if there exists a node $h \in H$ such that sequence σ_i of every player i equals to $\mathsf{seq}_i(h)$. Now, the equilibrium realization plans can be computed using the following feasibility LCP (e.g., see (Shoham and Leyton-Brown 2009) p. 135):

$$v_{\mathsf{inf}_i(\sigma_i)} = s_{\sigma_i} + \sum_{I'_i \in \mathcal{I}_i : \mathsf{seq}_i(I'_i) = \sigma_i} v_{I'_i} + \sum_{\sigma_{-i} \in \Sigma_{-i}} g_i(\sigma_i, \sigma_{-i}) \cdot r_{-i}(\sigma_{-i}) \qquad \forall i \in N \; \forall \sigma_i \in \Sigma_i \qquad (2)$$

$$r_i(\emptyset) = 1 \qquad \forall i \in N \qquad (3)$$

$$r_i(\sigma_i) = \sum_{a \in A(I_i)} r_i(\sigma_i a) \qquad \forall i \in N \; \forall I_i \in \mathcal{I}_i, \; \sigma_i = \mathsf{seq}_i(I_i) \qquad (4)$$

$$0 = r_i(\sigma_i) \cdot s_{\sigma_i} \qquad \forall i \in N \; \forall \sigma_i \in \Sigma_i \qquad (5)$$

$$0 \leq r_i(\sigma_i) \; ; \qquad 0 \leq s_{\sigma_i} \qquad \forall i \in N \; \forall \sigma_i \in \Sigma_i \qquad (6)$$

Constraints (2) ensure that the expected value v_{I_i} in each information set of player i equals to the value of the best response in this information set for player i against the strategy of the opponent $-i$ (i.e., for each action applicable in information set I_i, v_{I_i} is greater than sum of all expected values of information sets and leafs reachable after playing an action in this information set). Note that in our formulation we represent the inequality as an equality with slack variables s_{σ_i}; there is one positive slack variable for each sequence of each player σ_i. We use this representation as it would be convenient for our novel algorithms to operate with these constraints by using the slack variables.

Constraints (3-4) ensure that the realization plans of both players satisfy the network-flow constraints: the probability of reaching information set I_i using sequence $\sigma_i = \mathsf{seq}_i(I_i)$ must be equal to the sum of probabilities of sequences extended by actions a defined in I_i ($a \in A(I_i)$).

Finally, there are complementary slackness conditions (constraints (5)) stating that a sequence σ_i is either never played (i.e., $r_i(\sigma_i) = 0$) or slack variable $s_{\sigma_i} = 0$, which ensures that sequence σ_i is part of the best response for player i ensured by the constraint (2).

Existing Algorithms for Computing SSE

We now focus on the Stackelberg setting and fix the roles of the two players. Player 1 is the leader that commits to a strategy. Player 2 is the follower that plays the best response to the strategy of the leader. The baseline algorithm for computing SSE in single-step normal-form games (NFGs), introduced by (Conitzer and Sandholm 2006), is based on solving multiple linear programs – for every pure strategy of the follower $\pi_2 \in \Pi_2$ we can compute a mixed strategy for the leader $\delta_1^{\pi_2}$ such that (1) playing π_2 is the best response of the follower against $\delta_1^{\pi_2}$ (i.e., $BR_2(\delta_1^{\pi_2}) = \pi_2$) and (2) $\delta_1^{\pi_2}$ is such that it maximizes expected utility of the leader. Such a mixed strategy $\delta_1^{\pi_2}$ can be found using linear program; hence, the baseline algorithm calculates $|\Pi_2|$ linear programs, and for each pure strategy of the leader the algorithm computes $\delta_1^{\pi_2}$ and the expected utility of the leader $u_1(\delta_1^{\pi_2}, \pi_2)$. Finally the algorithm selects such strategy profile $(\delta_1^{\pi_2^*}, \pi_2^*)$ for which the expected utility u_1 is maximal. Often, certain pure strategies of the follower can never be best responses (e.g., strictly dominated strategies), in which case the LPs for these pure strategies are not feasible.

Follow-up works primarily focused on the Bayesian setting, where the leader is playing against one of possible followers with different preferences. Although the work (Conitzer and Sandholm 2006) showed that finding SSE in Bayesian games is NP-hard, new algorithms were able to scale to real-world scenarios. First successful algorithm was based on mixed-integer linear programming (MILP) introduced in (Paruchuri et al. 2008). The main advantage of the MILP formulation is in avoiding the exponential Harsanyi transformation of a Bayesian game into a normal-form game. New algorithms that followed were inspired by column/constraint generation techniques that restrict the number of linear programs to be solved in multiple LPs approach exploiting hierarchical consideration of the types of the follower (Jain, Tambe, and Kiekintveld 2011), and more tight calculation of bounds by using convex hull relaxation and Bender's decomposition in (Yin and Tambe 2012). Large volume of works focused on more specific game models including security games (Kiekintveld et al. 2009; Tambe 2011), or patrolling games (Basilico, Gatti, and Amigoni 2009; Vorobeychik, An, and Tambe 2012).

Described baseline algorithms could be, in principle, applied also for solving EFGs – we can transform any EFG into a NFG using the concept of pure and mixed strategies. However, since the number of pure strategies is exponential, the baseline algorithm would have to solve exponentially many LPs (one LP for each pure strategy of the follower). Moreover, to ensure that the currently fixed strategy of the follower is the best response, each LP would be of an exponential size (exponentially many constraints, one constraint for each pure strategy of the follower). Therefore, the scalability of such approach is very limited.

In the next section we therefore introduce a new algorithm that builds on the existing algorithms for computing SSE, however, that directly exploits the compact representation of strategies using the sequence form.

Sequence-Form for SSE in EFGs

We first describe the modification of the baseline algorithm for computing Strong Stackelberg Equilibrium (SSE) based on solving multiple LPs. We introduce a novel LP that for a fixed pure strategy of the follower π_2 computes a mixed strategy δ_1 satisfying 2 conditions required for the SSE: (1) π_2 is the best response to δ_1 and (2) δ_1 is such that the expected utility for the leader is maximal. Afterwards we extend this LP and formulate a MILP for computing SSE in EFGs. For each algorithm, we first describe the key ideas leading to the formulation, following by the complete listing of all the constraints of the mathematical programs.

Multiple Sequence-Form LPs for SSE

Our algorithm exploits the compact sequence form and represents the strategy of both players as realization plans. First, the realization plan of the leader r_1 needs to satisfy the network-flow constraints (3-4). Next, let us denote Σ_2^{BR} sequences of the follower that correspond to the pure realization plan of the follower (i.e., the sequences that are played with probability 1 in the fixed pure realization plan of the follower π_2). The algorithm needs to ensure that the realization plan of the leader is such that the sequences in Σ_2^{BR} are the best response to r_1. To do this, the algorithm exploits the constraint (2); there will be one such constraint for each sequence of the follower $\sigma_2 \in \Sigma_2$. To ensure that sequences from Σ_2^{BR} form the best response, we strictly set slack variables s_{σ_2} to be equal to zero for sequences in Σ_2^{BR}. Such tightening of the slack variables causes the expected utility value for a certain information set to be equal to the expected utility gained by playing actions corresponding to sequences in Σ_2^{BR}. More precisely, consider a sequence $\sigma_2 \in \Sigma_2^{BR}$, where the last action of the sequence, a, is applicable in information set $I = \mathsf{inf}_2(\sigma_2)$ (i.e., $\exists \sigma_2' \in \Sigma_2^{BR}$ such that $\sigma_2 = \sigma_2' a$). Now, if a slack variable s_{σ_2} is forced to be zero, then it holds

$$v_I = \sum_{I_2' \in \mathcal{I}_2 : \mathsf{seq}_2(I_2') = \sigma_2} v_{I_2'} + \sum_{\sigma_1 \in \Sigma_1} g_2(\sigma_2, \sigma_1) \cdot r_1(\sigma_1) \qquad (7)$$

For any other action applicable in I (i.e., $\forall b \in A(I)\ a \neq b$) the constraints for sequences $\sigma_2' b$ have a positive slack variable. Therefore, the realization plan of the leader r_1 must be such that the expected value of the right side in this constraint gained by actions b must be less or equal to v_I because the expected value of the right side of the constraint can be only increased with a positive slack variable. This restriction, however, corresponds to the fact that action a is best to be played in information set I. Since this holds for all sequences in Σ_2^{BR} and these sequences correspond to a pure realization plan for the follower, the instantiations of constraints (2) restrict the strategy of the leader r_1 such that Σ_2^{BR} is the best response of the follower.

Finally, we need to specify the objective function. Since we have variables representing the realization plan of the leader r_1 and fixed set of sequences of the opponent Σ_2^{BR}, we can simply calculate an expected utility for the leader using function g_1 only for the sequences in Σ_2^{BR}. We thus arrive to the final formulation of the linear program for computing the desired realization plan for the leader:

$$\max_{r_1, v_I, s_\sigma} \sum_{\sigma_1 \in \Sigma_1} \sum_{\sigma_2 \in \Sigma_2^{BR}} r_1(\sigma_1) g_1(\sigma_1, \sigma_2) \qquad (8)$$

$$v_{\mathsf{inf}_2(\sigma_2)} = s_{\sigma_2} + \sum_{I' \in \mathcal{I}_2 : \mathsf{seq}_2(I') = \sigma_2} v_{I'} + \sum_{\sigma_1 \in \Sigma_1} r_1(\sigma_1) g_2(\sigma_1, \sigma_2) \qquad \forall \sigma_2 \in \Sigma_2 \qquad (9)$$

$$r_1(\emptyset) = 1 \qquad (10)$$

$$0 \leq r_1(\sigma_1) \qquad \forall \sigma_1 \in \Sigma_1 \qquad (11)$$

$$r_1(\sigma_1) = \sum_{a \in A(I_1)} r_1(\sigma_1 a) \qquad \forall I_1 \in \mathcal{I}_1, \sigma_1 = \mathsf{seq}_1(I_1) \qquad (12)$$

$$0 \leq s_{\sigma_2} \qquad \forall \sigma_2 \in \Sigma_2 \qquad (13)$$

$$0 = s_{\sigma_2} \qquad \forall \sigma_2 \in \Sigma_2^{BR} \qquad (14)$$

Note that the sequences in Σ_2^{BR} must form a complete pure realization plan of the follower; hence, the set must be non-empty (empty sequence is always in Σ_2^{BR}) and every sequence in Σ_2^{BR} must also have a continuation sequence in Σ_2^{BR} if possible. Formally, if there is a sequence $\sigma_2 \in \Sigma_2^{BR}$ and there exists an information set $I \in \mathcal{I}_2$ such that $\mathsf{seq}_2(I) = \sigma_2$ then there must exist exactly one sequence $\sigma_2' \in \Sigma_2^{BR}$ such that $\mathsf{inf}_2(\sigma_2') = I$. This condition ensures that for every information set reachable under the assumption the follower is playing realization plan corresponding to Σ_2^{BR}, there is at least one slack variable forced to be zero. Violating this condition causes the LP to return incorrect results since values v_I could be arbitrarily increased using slack variables without restricting the realization plan of the leader.

By exploiting the sequence-form representation we avoided the exponential number of constraints. Instead of having one constraint for each possible strategy (to ensure that currently selected is the best response), we have a single constraint for each sequence of the follower. The presented LP is thus of a linear size to the size of the game tree. However, the algorithm would still need to solve exponentially many of such LPs (one for each pure realization plan of the follower) to compute SSE. To further improve the algorithm, we can exploit the reformulation using mixed-integer linear programming (MILP).

Sequence-Form MILP for Computing SSE

Similarly to work in Bayesian games (Paruchuri et al. 2008), we can reformulate the problem of solving Stackelberg equilibrium as a MILP. Since the best response of the follower is only in pure strategies, we can represent the strategy of the follower as a pure realization plan with binary variables $r_2 : \Sigma_2 \to \{0, 1\}$. The realization plans are restricted as usual and the network-flow constraints (3-4) apply. We again use the same idea of enforcing certain slack variables s_{σ_2} to be equal to zero for the sequences used in the realization plan of the follower. However, since the realization plan of the follower is not fixed but it is represented with variables r_2, we need to force slack variables $s_{\sigma_2} = 0$ whenever $r_2(\sigma_2)$ equals to 1. Since r_2 are binary variables, we can use the following constraint to ensure this:

$$0 \leq s_{\sigma_2} \leq (1 - r_2(\sigma_2)) \cdot M \qquad (15)$$

where M is a large constant. Now, using the constraint (9) we restrict variables r_1 and r_2 such that $r_2 = BR_2(r_1)$.

Finally, we need to modify the objective function, since the strategy of the follower is no longer fixed. The main idea is to use a new variable p that semantically corresponds to the probability distribution over leafs in the game tree considered the strategy of both players. Such a probability corresponds to a multiplication of variables r_1 and r_2. However, since the r_2 are binary variables we do not need to use multiplication. Consider a leaf $z \in Z$ and assume sequences σ_1 and σ_2 lead to this leaf for the leader and the follower respectively (i.e., by executing the actions in these sequences, leaf z is reached). Then, the value $p(z)$ is either equal to $r_1(\sigma_1)$ (when the follower plays σ_2 with probability 1), or it is equal to 0 otherwise (the follower is not playing σ_2). We can ensure this behavior with a set of linear constraints (see below, constraints (21-23)). Putting all together, we arrive to the formulation of the problem of computing SSE for EFGs as a MILP (note, that the presented MILP has linear size to the size of the game tree, with only $|\Sigma_2|$ binary variables):

$$\max_{p,r,v,s} \sum_{z \in Z} p(z) u_1(z) \mathcal{C}(z) \tag{16}$$

$$v_{\mathsf{inf}_2(\sigma_2)} = s_{\sigma_2} + \sum_{I' \in \mathcal{I}_2 : \mathsf{seq}_2(I') = \sigma_2} v_{I'} + \sum_{\sigma_1 \in \Sigma_1} r_1(\sigma_1) g_2(\sigma_1, \sigma_2) \quad \forall \sigma_2 \in \Sigma_2 \tag{17}$$

$$r_i(\emptyset) = 1 \qquad \forall i \in N \tag{18}$$

$$r_i(\sigma_i) = \sum_{a \in A_i(I_i)} r_i(\sigma_i a) \quad \forall i \in N \; \forall I_i \in \mathcal{I}_i, \sigma_i = \mathsf{seq}_i(I_i) \tag{19}$$

$$0 \leq s_{\sigma_2} \leq (1 - r_2(\sigma_2)) \cdot M \qquad \forall \sigma_2 \in \Sigma_2 \tag{20}$$

$$0 \leq p(z) \leq r_2(\mathsf{seq}_2(z)) \qquad \forall z \in Z \tag{21}$$

$$0 \leq p(z) \leq r_1(\mathsf{seq}_1(z)) \qquad \forall z \in Z \tag{22}$$

$$1 = \sum_{z \in Z} p(z) \tag{23}$$

$$r_2(\sigma_2) \in \{0, 1\} \qquad \forall \sigma_2 \in \Sigma_2 \tag{24}$$

$$0 \leq r_1(\sigma_1) \leq 1 \qquad \forall \sigma_1 \in \Sigma_1 \tag{25}$$

Experiments

We now turn to the experimental evaluation to demonstrate the scalability of our algorithm. We evaluate both variants of our algorithm – first variant with multiple LPs (denoted MULTILP), and the second variant with MILP – against the baseline approaches based on the transformation of EFGs into the normal form, denoted as EXPMULTILP (Conitzer and Sandholm 2006) and DOBBS (Paruchuri et al. 2008).

The main factor determining the complexity of the game is the size of the strategy space of the follower, since it plays a crucial part in the algorithms – one LP is solved for each pure realization plan of the follower in MULTILP, and the number of binary variables depends on the number of sequences of the follower in MILP. We analyze the scalability of the algorithms in two different settings: (1) a security-inspired search game (inspired by (Bosansky et al. 2013)) with a very large strategy space of the leader, while the strategy space of the follower is very small, and (2) randomly generated extensive-form games. While the first scenario allows us to scale to large game trees, the strategy space is more balanced in the second scenario and we analyze the scalability with respect to the increasing number of realization plans of the follower.

Experiment Settings

We use a domain-independent implementation of all algorithms and IBM CPLEX 12.5 for solving LPs and MILPs.

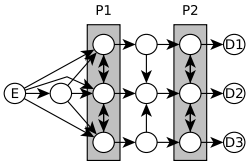
Search Game The search game is played on a directed graph (see Figure 1). The follower aims to reach one of the destination nodes (D1 – D3) from starting node (E), while the leader aims to encounter the follower with one of the two units operating in the shaded areas of the graph (P1 and P2). The follower receives different reward for reaching different destination node (the reward is randomly selected from interval $[1, 2]$). The leader receives positive reward 1 for capturing the follower. The follower leaves tracks in the visited nodes that can be discovered if the leader visits the node later in the game, but the follower can decide to erase the tracks in the current node (it takes one turn of the game).

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. Observation is a number from a limited 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 $\mathsf{seq}_i(h)$ and the observations generated by the actions of the opponent $-i$ belong to the same information set. We generate the games of differing sizes by varying parameters $\max_A = \{3, 4, 5\}$, $\max_O = \{2, 3, 4\}$, and depth of the game (up to 5 actions for each player). The utility values for players in leafs are assigned randomly from interval $[-5, 5]$. Finally, we are also interested in the effects of the correlation between the utility values on the performance of the algorithms (i.e., the utility values are generated to have a certain correlation factor; correlation -1 corresponds to zero-sum games, 1 to identical utility functions).

Results

Search Game The results on the search game (see the table in Figure 1) show that our algorithm outperforms the baseline approaches by several orders of magnitude. We scale the game by increasing the number of steps in the game. All algorithms were able to find the solution for 5 steps – it took EXPMULTILP more than 2 hours to compute the solution, while our MILP solved the game in slightly over 11 minutes. Making another step was possible only for our MILP that solved the largest instance in 6.3 hours. The size of this instance was 8.6×10^5 nodes with 470 pure realization plans of the follower. This results demonstrate the ability of using our MILP algorithm in sequential security scenarios where a large strategy space of the leader, caused by deploying and scheduling multiple resources, is more common.

Randomly Generated Games The results on random games confirm the dominance of our algorithm (see Figure 2). The left graph compares computation times of our



Algs \ Steps	4	5	6
EXPMULTILP	89	7,764	-
DOBBS	122	3,067	-
MULTILP	41	2,485	-
MILP	19	676	22,847

Figure 1: (Left) Graph for the search game, where the leader operates two units in the shaded areas, while the follower is planning to cross the area unobserved. (Right) Computation times (in seconds) for the search game with increasing number of steps in the game.

variants with the baseline approaches. Reported values are means out of 20 games, the standard error was always very small compared to the mean value and the differences between the algorithms (not visible in the graph). The results show that even a slight increase in the number of realization plans causes the baseline approaches to perform extremely slowly (note the logarithmic y-scale). Even relatively small games with 4000 realization plans of the follower take EXPMULTILP more than 80 minutes to solve, while our MILP solves such games in 225 miliseconds.

Therefore we further scaled the games comparing only our variants (the right graph in Figure 2, note that both scales are logarithmic). The depicted results are means of at least 100 different games, the standard error was again marginal. The results show that MILP is on average at least a magnitude faster compared to the MULTILP variant. The largest instances MILP was able to solve contained more than 10^7 realization plans of the follower taking up to 2 days to solve.

The reported results are for the correlation factor set to -0.5. When the correlation factor is decreased, it is more difficult for the leader to maximize the utility, which is strongly (but not absolutely) negatively correlated to the utility of the follower. With the correlation set to -0.8 the computation time of our MILP increases to 410ms for 4000 realization plans. On the other hand, increasing the correlation factor makes the games easier to solve (games with the correlation set to -0.2 are solved in 193ms by our MILP).

Discussion

The experimental evaluation shows that our novel algorithm dramatically outperforms the baseline approaches based on the transformation of EFGs into the normal form. This advantage is not surprising, however, the experiments show that our MILP algorithm is able to scale to moderately large sequential games with up to 10^5 nodes, or more than 10^6 pure realization plans of the follower, which can be sufficient for some real-world scenarios.

The comparison between the two variants of our algorithm is strongly in favor of the MILP variant. The bottleneck of the MULTILP variant is the evaluation of each LP for every pure realization plan of the follower, since the algorithm currently does not use any pruning techniques, or any iterative methods (e.g., branch-and-price or column/constraint generation methods). Using such techniques is, however, the key behind the success of the state-of-the-art algorithms for solving Bayesian Stackelberg games (Jain,

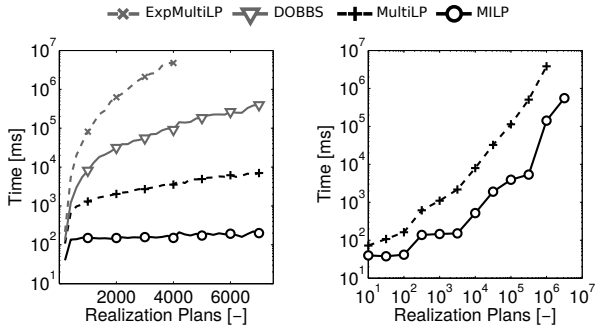


Figure 2: Comparison of the scalability of the exponential baseline approaches and the variants of our algorithm.

Tambe, and Kiekintveld 2011; Yin and Tambe 2012). Unfortunately, exploiting the same principles in EFGs is not straightforward due to a substantially different reason for the exponential number of the strategies of the follower.

The main challenge stems from computing an upper bound on the Stackelberg value for a partially instantiated strategy of the follower (the lower bound is provided by solving a LP for any complete strategy of the follower). Recall that described LPs do not work if the strategy of the follower is not a complete realization plan. In the Bayesian setting, a partially instantiated strategy of the follower corresponds to fixing the strategy for certain types of the follower. This provides a reasonable upper bound, since the remaining unfixed follower types can alter the value only with respect of their probability of appearing. The situation is different in EFGs. Partially fixing the strategy of the follower only restricts the achievable leafs in the game tree, but without further assumptions on the utility structure it does not restrict the possible Stackelberg value for the leader in general. A naïve upper bound, the best response of the leader, provides a very optimistic upper bound that does not improve the computation times of MULTILP enhanced with a branch-and-bound technique.

Conclusion

This paper presents a novel algorithm for computing Strong Stackelberg equilibria in extensive-form games (EFGs) by exploiting the compact sequence-form representation of strategies. We provide two formulations of our algorithm: one based on solving multiple linear programs (LPs), and one based on mixed-integer linear program. Our novel algorithm dramatically outperforms existing approaches based on the exponential transformation of EFGs into the normal form and allow us to solve significantly larger games.

Our work opens two important lines of research. First of all, the presented algorithm can be a basis for creating domain-dependent algorithms to solve large games, for example in the security domain. Second, our variant based on solving multiple LPs can be further improved by using iterative and decomposition techniques. To allow this, a new (or domain-specific) algorithm for computing a tight upper bound on the Stackelberg value in EFGs based on a partially instantiated strategy of the follower must be designed.

Acknowledgments

This research was supported by the Czech Science Foundation (grant no. P202/12/2054). Bosansky also acknowledges support from 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 the support from the Center for Research in Foundations of Electronic Markets (CFEM), supported by the Danish Strategic Research Council.

References

Basilico, N.; Gatti, N.; and Amigoni, F. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 57–64.

Bosansky, B.; Kiekintveld, C.; Lisy, V.; Cermak, J.; and Pechoucek, M. 2013. Double-oracle Algorithm for Computing an Exact Nash Equilibrium in Zero-sum Extensive-form Games. In *Proceedings of International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 335–342.

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.

Jain, M.; Tambe, M.; and Kiekintveld, C. 2011. Quality-bounded solutions for finite Bayesian Stackelberg games: Scaling up. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, 997–1004.

Karp, R. M. 1971. Reducibility among combinatorial problems. 85–103.

Kiekintveld, C.; Jain, M.; Tsai, J.; Pita, J.; Ordóñez, F.; and Tambe, M. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, 689–696.

Koller, D.; Megiddo, N.; and von Stengel, B. 1996. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior* 14(2):247–259.

Leitmann, G. 1978. On generalized stackelberg strategies. *Optimization Theory and Applications* 26:637–643.

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. New York, NY, USA: ACM.

Paruchuri, P.; Pearce, J.; Marecki, J.; Tambe, M.; Ordonez, F.; and Kraus, S. 2008. Playing games for security: an efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 895–902.

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 *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, 125–132.

Shieh, E.; An, B.; Yang, R.; Tambe, M.; Baldwin, C.; Direnzo, J.; Meyer, G.; Baldwin, C. W.; Maule, B. J.; and Meyer, G. R. 2012. PROTECT : A Deployed Game Theoretic System to Protect the Ports of the United States. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 13–20.

Shoham, Y., and Leyton-Brown, K. 2009. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press.

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

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

von Stackelberg, H. 1934. Marktform und gleichgewicht.

von Stengel, B., and Zamir, S. 2004. Leadership with commitment to mixed strategies.

von Stengel, B. 1996. Efficient computation of behavior strategies. *Games and Economic Behavior* 14:220–246.

Vorobeychik, Y.; An, B.; and Tambe, M. 2012. Adversarial patrolling games. In *Proceedings of the AAAI Spring Symposium*.

Yin, Z., and Tambe, M. 2012. A unified method for handling discrete and continuous uncertainty in bayesian stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 855–862.

Yin, Z.; Korzhyk, D.; Kiekintveld, C.; Conitzer, V.; and Tambe, M. 2010. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 1139–1146.