# Incremental Strategy Generation for Stackelberg Equilibria in Extensive-Form Games

JAKUB ČERNÝ, Czech Technical University in Prague, Czech Republic
BRANISLAV BOŠANSKÝ, Czech Technical University in Prague, Czech Republic
CHRISTOPHER KIEKINTVELD, University of Texas at El Paso, USA

Dynamic interaction appears in many real-world scenarios where players are able to observe (perhaps imperfectly) the actions of another player and react accordingly. We consider the baseline representation of dynamic games—the extensive form—and focus on computing Stackelberg equilibrium (SE), where the leader commits to a strategy to which the follower plays a best response. For one-shot games (e.g., security games), strategy-generation (SG) algorithms offer dramatic speed-up by incrementally expanding the strategy spaces. However, a direct application of SG to extensive-form games (EFGs) does not bring a similar speed-up since it typically results in a nearly-complete strategy space. Our contributions are twofold: (1) for the first time we introduce an algorithm that allows us to incrementally expand the strategy space to find a SE in EFGs; (2) we introduce a heuristic variant of the algorithm that is theoretically incomplete, but in practice allows us to find exact (or close-to optimal) Stackelberg equilibrium by constructing a significantly smaller strategy space. Our experimental evaluation confirms that we are able to compute SE by considering only a fraction of the strategy space that often leads to a significant speed-up in computation times.

CCS Concepts: • **Theory of computation** → **Algorithmic game theory**; **Exact and approximate computation of equilibria**;

Additional Key Words and Phrases: Extensive-form games; strong Stackelberg equilibrium; correlated equilibrium; strategy generation

## 1 INTRODUCTION

Many game-theoretic models inspired by real-world scenarios are dynamic, in that they model sequences of interacting moves and observations by the players. The players may have

This research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-13-2-0045 (ARL Cyber Security CRA). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes not with standing any copyright notation here on.
Access to computing and storage facilities owned by parties and projects contributing to the National Grid Infrastructure MetaCentrum provided under the programme "Projects of Large Research, Development, and Innovations Infrastructures" (CESNET LM2015042), is greatly appreciated.

Authors' addresses: Jakub Černý, Czech Technical University in Prague, Technická 2, Prague, 16627, Czech Republic, jakub.cerny@agents.fel.cvut.cz; Branislav Bošanský, Czech Technical University in Prague, Technická 2, Prague, 16627, Czech Republic, bosansky@fel.cvut.cz; Christopher Kiekintveld, University of Texas at El Paso, 500 West University Ave. El Paso, Texas, 79968-0518, USA, cdkiekintveld@utep.edu.


*ACM EC'18, June 18–22, 2018, Ithaca, MA, USA.* ACM ISBN 978-1-4503-5829-3/18/06…$15.00
https://doi.org/10.1145/3219166.3219219

uncertainty about the effects of actions and may receive only imperfect observations of the actions chosen by other players. The baseline formalism for reasoning about dynamic games with a limited horizon is *extensive form games* (EFGs), which can be represented using game trees. Many scenarios can be modeled as EFG, including card games like poker [5, 15], security games with patrols (e.g., examples in [4, 11]), computer network attacks [6, 7], and synthetic biology games in medicine [16].

The roles of the players in many real-world games are asymmetric. One player (*the leader*) has the power to commit to a strategy and the other player (*the follower*) plays a best response. For example, the leader can correspond to 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 Stackelberg Equilibrium (SE) [12, 22]. We follow the common assumption in the literature that the follower break ties in favor of the leader; hence, we compute a Strong Stackelberg Equilibrium (SSE)[1].

The problem of computing SSE in EFGs is known to be NP-complete [4, 13]. There are several existing algorithms for computing SSE in EFGs. Bošanský and Čermák [4] introduced a mixed-integer linear program (MILP) that extends the sequence form linear program for computing NE in EFGs to SSE [10, 20]. The scalability of this approach was improved by first computing a correlated version of the SSE (called Stackelberg Extensive-Form Correlated Equilibrium, SEFCE) and then using a search to refine the SEFCE into a SSE [19]. Finally, Kroer et al. [11] introduced a novel MILP formulation that incorporates interval uncertainty in the utility of the follower, as well as a limited lookahead approach assuming that the follower is not perfectly rational and can only reason to a limited depth.

We introduce a novel strategy-generation (SG) technique that starts from a small restricted game and incrementally expands the game tree of a two-player extensive-form game to compute SSE. The inspiration for the algorithm is based on previous algorithms that have been very successful for one-shot games (e.g., many types of security games [8, 9, 23]), as well as for zero-sum sequential [2, 14, 18] and EFGs [3]. However, none of these previous SG approaches translates directly to a practical algorithm for EFGs. In the first case, the main step in each iteration is to add sequences of actions of the leader that can potentially increase the objective of a (MI)LP. This is done by computing the reduced costs for variables that are not included in the program yet [8]. We investigate this approach for EFG in Section 3, but encounter a fundamental problem in that this approach adds many unnecessary sequences, leading to "reduced" games that are comparable in size to the original. The SG algorithm for zero-sum EFGs relies heavily on the zero-sum assumption and expands the game tree by adding best-response sequences into the game tree. Unfortunately, this approach does not converge to SSE in a general sum EFG.

We can now state the two main challenges we must solve to develop an effective SG algorithm for computing SSE in EFGs: *(1) What is the representation of the restricted game, and in particular, the abstracted parts of the game tree? (2) What is the methodology for expanding the restricted game?* We address both of these challenges and show that (1) the abstracted parts of the game tree can be effectively represented using a subset of pareto-optimal outcomes; (2) the expansion can be done when an outcome from the abstracted part of the game tree is used in the current solution of the restricted game. Our technique is independent and can be combined with any of the existing algorithms

[1] We expect that our general approach would be applicable for computing other variations of Stackelberg equilibrium, but leave this to future work.

for computing SSE. We use the SEFCE-based algorithm [19] since it has the best-known scalability. The LP for computing SEFCE is quadratic in the size of the game tree, so constructing a smaller game tree should also have a significant performance benefit. In addition to an exact algorithm, we also introduce heuristic variants that lose theoretical guarantees on convergence in exchange for much smaller games; our experimental results show that the heuristic variant can compute exact Stackelberg equilibrium of games with more than $10^7$ states by constructing only 8% of the original SEFCE LP. More aggressive heuristics are able to find near-optimal solutions considering only 3% of the LP, while the outcome for the leader is on 6.38% worse compared to the optimum. Often, such a dramatic reduction in the size of the LP leads to significant speed-up in computation times.

## 2 EXTENSIVE-FORM GAMES AND STACKELBERG SOLUTION CONCEPTS

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 $i$ ($u_i : \mathcal{Z} \to \mathbb{R}$). A 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 an 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 the 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* $\Pi_i$ assign one action for each $I \in \mathcal{I}_i$. A more efficient representation in the form of *reduced pure strategies* $\Pi_i^*$ 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 $\Pi_i$. 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 $\delta$. A *best response* of player $i$ to the opponent's strategy $\delta_{-i}$ 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 [10]. 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 $\sigma_i$ is a *prefix* of $\sigma_i'$ ($\sigma_i \sqsubseteq \sigma_i'$) if $\sigma_i'$ is obtained by finite number (possibly zero) of extensions of $\sigma_i$. We use $seq_i(I_i)$ and $seq_i(h)$ to denote the sequence of $i$ leading to $I_i$ and $h$, respectively. We use the function $inf_i(\sigma_i')$ to obtain the information set in which the last action of the sequence $\sigma_i'$ is taken. For an empty sequence, function $inf_i(\emptyset)$ returns the information set of the root node. A mixed strategy of a player can now be

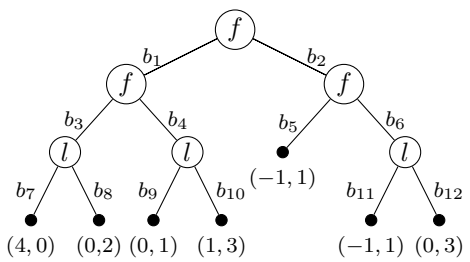
represented as a *realization plan* ($r_i : \Sigma_i \to \mathbb{R}$). A realization plan for a sequence $\sigma_i$ is the probability that player $i$ will play $\sigma_i$ under the assumption that the opponent plays to allow the actions specified in $\sigma_i$ to be played. By $g_i : \Sigma_l \times \Sigma_f \to \mathbb{R}$ we denote the *extended utility function*, $g_i(\sigma_l, \sigma_f) = \sum_{z \in \mathcal{Z} | seq_l(z) = \sigma_l \wedge seq_f(z) = \sigma_f} u_i(z) C(z)$. If no leaf is reachable with a pair of sequences $\sigma$, the value of $g_i$ is 0.

***Stackelberg Solution Concepts in EFGs.*** We provide a formal definition of Strong Stackelberg Equilibrium (SSE) (e.g., in [12]) and Stackelberg Extensive-Form Correlated Equilibrium (SEFCE) [1, 13] and give the intuition on an example game.

*Definition 2.1.* A strategy profile $\delta = (\delta_l, \delta_f)$ is a *Strong Stackelberg Equilibrium* if $\delta_l$ is an optimal strategy of the leader given that the follower best-responds. Formally:

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

The SSE of the game in Figure 1[2] (the first utility in every leaf is for the leader, second for the follower) prescribes the leader to commit to playing actions $b_7$ and $b_{10}$ in the information sets in the left subtree and $b_{11}$ in the right subtree. The strategy of the follower is then to play $b_1$ in the root of the game and $b_4$ in the left subtree, leading to the expected utility of 1 for the leader.




| | $\{b_1, b_3\}$ | $\{b_1, b_4\}$ | $\{b_2, b_5\}$ | $\{b_2, b_6\}$ |
|---|---|---|---|---|
| $\{b_7, b_9, b_{11}\}$ | 0.25 | 0 | 0 | 0 |
| $\{b_7, b_9, b_{12}\}$ | 0 | 0 | 0 | 0 |
| $\{b_7, b_{10}, b_{11}\}$ | 0 | 0.5 | 0 | 0 |
| $\{b_7, b_{10}, b_{12}\}$ | 0 | 0 | 0 | 0 |
| $\{b_8, b_9, b_{11}\}$ | 0.25 | 0 | 0 | 0 |
| $\{b_8, b_9, b_{12}\}$ | 0 | 0 | 0 | 0 |
| $\{b_8, b_{10}, b_{11}\}$ | 0 | 0 | 0 | 0 |
| $\{b_8, b_{10}, b_{12}\}$ | 0 | 0 | 0 | 0 |

| | $\emptyset$ | $b_7$ | $b_8$ | $b_9$ | $b_{10}$ | $b_{11}$ | $b_{12}$ |
|---|---|---|---|---|---|---|---|
| $\emptyset$ | 1 | 1 | 0 | 0.5 | 0.5 | 0 | 0 |
| $b_1$ | 0.75 | 0.75 | 0 | 0.25 | 0.5 | - | - |
| $b_2$ | 0.25 | 0.25 | 0 | 0.25 | 0 | - | - |
| $b_1 b_3$ | 0.5 | 0.5 | 0 | 0.5 | 0 | - | - |
| $b_1 b_4$ | 0.5 | 0.5 | 0 | 0 | 0.5 | - | - |
| $b_2 b_5$ | 1 | 1 | 0 | - | - | 0 | 0 |
| $b_2 b_6$ | 0 | 0 | 0 | - | - | 0 | 0 |

Fig. 1. (Left) An EFG with different SEFCE and SSE. Each internal node is labeled by a player who acts in this node, while under every terminal node is a tuple of utilities obtained by the first player and the second player, respectively. Every edge is labeled by an action performed on a way from the node above to the node below. The direction of the edges is omitted, but the tree is assumed to be traversed from top to bottom. (Right Up) The SEFCE distribution over $\Pi^*$. (Right Down) The SEFCE correlation plan.

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.

[2]The example is from [19].

*Definition 2.2.* A probability distribution $\lambda$ on reduced pure strategy profiles $\Pi^*$ 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 $\tilde{a}$ according to $\lambda$ such that the follower cannot gain by unilaterally deviating from $\tilde{a}$ 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.

SEFCE of the EFG in Figure 1 is shown in the top table, with the rows being labeled by the leader's strategies $\Pi_l^*$ and the columns by the follower's strategies $\Pi_f^*$). At the beginning of the game, the leader decides to send either $\{b_1, b_3\}$ or $\{b_1, b_4\}$ as a recommendation to the follower according to the probabilities in the table. In both cases, the follower commits to playing action $b_{11}$ in order to incentivize the follower to play $b_1$. If the follower receives $b_3$, the leader plays action $b_9$ and mixes uniformly between $b_7$ and $b_8$ in his right-most information set. If the follower is recommended $b_4$, the leader commits to playing $b_7$ and $b_{10}$. The expected utility of the leader in this equilibrium is 1.5, which is strictly better than in the SSE.

***Using SEFCE for computing SSE.*** The correlated variant of the Stackelberg equilibrium can be used for computing SSE. The main idea is to find SEFCE and then iteratively add constraints so the recommendations for the follower are such that in each information set, the follower can receive only a single action to be played as a recommendation. In that case, the follower is playing a pure best response and the strategy of the leader is the same as in the SSE.

We now describe the linear program for computing SEFCE. In this linear program, the strategies of the players are represented using a correlation plan for relevant sequences. The sequences are termed *relevant* when decisions at the information sets reachable by one of the sequences can affect the decisions at the information sets reachable by the other sequence.

*Definition 2.3 (Relevant sequences [21]).* A pair of sequences $(\sigma_1, \sigma_2)$ is termed *relevant* if and only if $\exists i \in \{1, 2\}$ either $\sigma_i = \emptyset$ or $\exists h, h' \in H, h' \sqsubseteq h; \sigma_i = seq_i(h) \wedge \sigma_{-i} = seq_{-i}(h')$.

The set of sequences of player $-i$ which form a relevant pair with $\sigma_i$ is denoted $rel(\sigma_i)$. For the EFG depicted in Figure 1 $rel(b_1) = rel(b_2) = \Sigma_l$, $rel(b_2b_5) = rel(b_2b_6) = \{\emptyset, b_{11}, b_{12}\}$, and $rel(b_1b_3) = rel(b_1b_4) = \{\emptyset, b_7, b_8, b_9, b_{10}\}$. Now it is possible to define a generalization of a realization plan suitable for joint probabilities of pairs of sequences, called a *correlation plan*. We have to ensure that in mutually relevant information sets the consistency of recommendations cannot be violated.

*Definition 2.4 (A correlation plan [21]).* A partial function $p : \Sigma_1 \times \Sigma_2 \to \mathbb{R}$ is a correlation plan if there is a probability distribution $\lambda$ on the set of reduced strategy profiles $\Pi^*$ so that for each relevant sequence pair $(\sigma_1, \sigma_2)$, the term $p(\sigma_1, \sigma_2)$ equals to $p(\sigma_1, \sigma_2) = \sum_{(\pi_1, \pi_2) \in \Pi^*} \lambda(\pi_1, \pi_2)$ where $\pi_1, \pi_2$ prescribe playing all of the actions in $\sigma_1$ and $\sigma_2$, respectively.

The correlation plans describe a joint realization probability $p(\sigma_l, \sigma_f)$ that a sequence $\sigma_f$ is recommended to the follower, in case the leader plays sequence $\sigma_l$. The bottom table of Figure 1 represents the correlation plan of the SEFCE strategies. The rows of the table are labeled by $\Sigma_f$, while columns are labeled by $\Sigma_l$. In every row identified by $\sigma_f$ is depicted the probability of the leader playing the corresponding column sequence in case the follower is recommended the sequence $\sigma_f$ and follows his recommendations. The irrelevant pairs of

sequences are denoted '-'. The correlation plan of every relevant pair $(\sigma_l, \sigma_f)$ is the sum of all probabilities of pure strategies containing actions from $\sigma_l$ and $\sigma_f$ in the top table.

In [21] the authors proved that correlation plans are sufficient to characterize the set of extensive-form correlated equilibrium (EFCE) in two-player games with no chance nodes. In [19] the authors used this characterization to formulate the following linear program computing SEFCE.

THEOREM 2.5 (STACKELBERG EXTENSIVE-FORM CORRELATED EQUILIBRIUM IN TWO–PLAYER GAME WITHOUT CHANCE MOVES [19]). *The distribution $\lambda$ on $\Pi^*$ defines a SEFCE if and only if $\lambda$ is a solution of the following linear program that maximizes leader's expected utility*

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

*and the respective correlation plan $p$ satisfies*

$$p(\emptyset, \emptyset) = 1; \quad 0 \leq p(\sigma_l, \sigma_f) \leq 1 \tag{3}$$

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

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

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

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

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

The first three constraints enforce the consistency of the correlation plan. The following constraint ensures that $v_{\sigma_f}$ is a representation of an expected payoff of the follower when he plays $\sigma_f$, assuming he follows his recommendations. The constraint consists of two parts – the first sum computes the expected utility of the leafs reached by playing according to $\sigma_l$ and $\sigma_f$, the second sum adds the contribution of the expected utility of information sets reachable by all the extensions of $\sigma_f$. The next constraint guarantees that the expected payoff $v(I, \sigma_f)$ is the maximum over all possible sequences leaving the information set $I$ (denoted as $seq_f(I)a$ for all possible actions $a \in A(I)$) after the follower is recommended to play $\sigma_f$. Finally, the last constraint forces the move which is recommended to the follower in the information set $I$ to be optimal. The value of SEFCE is always greater or equal than the value of SSE.

THEOREM 2.6 ([19]). *Assume a solution of the LP as described in Theorem 2.5. The objective value is greater than or equal to the expected utility of the leader in SSE.*

As stated before, SSE can be reached by a branch-and-bound algorithm [19] so that the recommendations for the follower are unique – the authors define *inconsistent recommendations* that are then fixed by a branch-and-bound type of search algorithm (BnB).

*Definition 2.7 (Inconsistent recommendations [19]).* 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, $\exists a, a' \in A(I), a \neq a', \exists \sigma_l, \sigma'_l \in \bigcup_{h \in I} seq_l(h)\ p(\sigma_l, seq_f(I)a) > 0 \wedge p(\sigma'_l, seq_f(I)a') > 0$. If there exists no such information set we say that $p$ uses only *consistent recommendations*.

**ALGORITHM 1:** BnB-Based Algorithm for computing SSE.

**Input:** An UB-SSE-LP $P$
**Output:** leader's expected utility and strategy profile in SSE
$M \leftarrow \{(\infty, \emptyset)\}$; $LB \leftarrow -\infty$; $p_c \leftarrow \emptyset$
**while** $M \neq \emptyset$ **do**
  $(UB, m) \leftarrow \max(M)$
  **if** $UB < LB$ **then**
    **return** $(LB, p_c)$
  apply($m$, $P$)
  **if** feasible($P$) **then**
    $(value, p) \leftarrow$ solve($P$)
    $\mathcal{I}_{in} \leftarrow$ inconsistentRecommendations($p$)
    **if** $\mathcal{I}_{in} = \emptyset$ **then**
      **if** $value > LB$ **then** $LB \leftarrow value$; $p_c \leftarrow p$
    **else** addRestrictions($(UB, m)$, $M$, $\mathcal{I}_{in}$, $value$)
  revert($m$, $P$)
**return** $(LB, p_c)$

The Algorithm 1 can be decomposed into iterative applications of the following steps: (1) solve an LP $P$ in line 2, (2) find the set of information sets of the follower with inconsistent recommendations $\mathcal{I}_{in}$ in line 1, and (3) restrict the leader's strategy to use only consistent recommendations in $\mathcal{I}_{in}$ by adding new constraints $m$ to the LP $P$ in line 1. The restrictions are added recursively until a restricted LP for SEFCE uses only consistent correlation plan $p$. At the beginning, the algorithm is initiated with an LP computing SEFCE.

## 3 CHALLENGES IN STRATEGY GENERATION FOR COMPUTING SSE IN EFGS

We now describe the general framework of a strategy generation (SG) technique for computing a Strong Stackelberg Equilibrium (SSE) in extensive-form games (EFGs). We formally define a restricted game as a subgraph of the original game tree and describe the baseline SG algorithm that extends the algorithm for one-shot games [8]. However, we show on a simple example that such a direct adaptation can easily generate the full game tree.

### 3.1 Strategy Generation for SSE Using Reduced Costs

The main idea of the previous approach [8] is to decompose the mathematical program for computing SSE into a master problem and a slave problem. In the master problem, we solve for SSE of a smaller restricted game (RG), where the RG is a subset of the original unrestricted game. In the slave problem, we search for such strategies of the leader to be added into the RG which have the largest positive impact on the objective, given the solution of the master problem. The impact is measured using *reduced costs*, calculated from the dual solution.

We translate this idea for computing SSE in EFGs while exploiting SEFCE solution concept. Therefore, our master program corresponds to the LP formulation of SEFCE (LP

in Theorem 2.5) and the slave problem identifies which strategies of the leader should be added to the restricted game.

After the SEFCE is found, the BnB algorithm is called in order to compute SSE. The only change from the original version depicted in Algortihm 1 is that it solves for SEFCE in the restricted game and after adding every restriction, the algorithm checks whether the RG should be expanded.

***Iterative Construction of SEFCE LP.*** We now detail how we translate this idea for computing SSE in EFGs. First, our master program – the LP formulation of SEFCE – will be solved for a restricted game that can be fully specified by the subset of sequences of the leader (called *allowed sequences*). Similarly to [3], we define the sets of nodes, actions, and information sets as subsets of the original unrestricted sets based on the allowed sequences. The RG is denoted as $G' = (\mathcal{N}, \mathcal{H}', \mathcal{Z}', \mathcal{A}', u, \mathcal{C}, \mathcal{I}')$ and the set of all sequences in the RG is denoted $\Sigma'$.

We assume the RG is always closed on prefixes of leader's sequences, such that for any leader's sequence in RG, all follower's relevant sequences are in RG. The leader's sequences hence fully define the RG. The set $\mathcal{H}'$ can be derived as

$$\mathcal{H}' \leftarrow \{h \in \mathcal{H} : \forall i \in \mathcal{N}\ seq_i(h) \in \Sigma'\}, \tag{9}$$

and the restriction $\mathcal{A}'$ of $\mathcal{A}$ to the sequences in the RG is constructed as

$$\mathcal{A}'(h) \leftarrow \{a \in \mathcal{A}(h) : ha \in \mathcal{H}'\} \quad \forall h \in \mathcal{H}'. \tag{10}$$

The leafs in the RG are $\mathcal{Z}' = \mathcal{Z} \cap \mathcal{H}'$. Finally, the information sets of the RG are derived as

$$\mathcal{I}'_i \leftarrow \{I_i \in \mathcal{I}_i : \exists h \in \mathcal{H}' \cap I_i\} \quad \forall i \in \mathcal{N}. \tag{11}$$

In the slave problem, we search for the leader's sequences not contained in the RG. The dual solution as for computing the reduced costs follows:

$$rc(\sigma_l) = \max_{\sigma_f \in rel(\sigma_l)} rc(p(\sigma_l, \sigma_f)), \tag{12}$$

where

$$\begin{aligned} rc(p(\sigma_l, \sigma_f)) \ = g_l(\sigma_l, \sigma_f) &- \sum_{I_l \in \mathcal{I}'_l : seq(I_l) = \sigma_l} d_4(I_l, \sigma_f) + d_4(inf(\sigma_l), \sigma_f) - \\ &- \sum_{I_f \in \mathcal{I}'_f : seq(I_f) = \sigma_f} d_5(I_f, \sigma_l) + d_5(inf(\sigma_f), \sigma_l) - d_6(\sigma_f) + \\ &+ \sum_{I_f \in \mathcal{I}'_f : \sigma_f \in \bigcup_{h \in I} rel(seq_l(h))} \ \sum_{a \in \mathcal{A}'(I_f)} g_f(\sigma_l, seq_f(I_f)a) d_7(I_f, \sigma_f, a), \end{aligned} \tag{13}$$

and the $d_j(\dots)$, $j \in \{4, 5, 6, 7\}$, are the dual values of the constraints 4, 5, 6 and 7 in the current solution of the LP computing SEFCE. In order for the algorithm to guarantee a convergence to an optimum, the RG does not contain any temporary leafs, in contrast to [3]. After enumerating the costs for all leader's sequences outside the RG, the sequences with the positive reduced costs $\hat{\Sigma}$ are added to the RG. Simultaneously, we include also all relevant sequences for the follower. The sequences $\Sigma'$ hence change as

$$\Sigma' \leftarrow \Sigma' \cup \hat{\Sigma} \cup rel\left(\hat{\Sigma}\right). \tag{14}$$

The LP for the restricted game is then extended by generating new correlation plan constraints 4 and 5; and constraints 6, 7 and 8 in case a new follower's sequence enter the RG. The

new correlation pairs $p(\sigma_l, \sigma_f)$ are added into the already existing constraints and to the objective.

At the beginning of the algorithm, the initial restricted game is constructed using the sequences forming an arbitrary reduced pure strategy of the leader. In practice, we started from the left-most branch of the game tree and continued adding the sequences of the leader until we arrived to the full reduced pure strategy. The process of solving the master problem and the slave problems is then repeated until all remaining leader sequences out of the RG have a non-positive cost.

### 3.2 Limitations of Reduced Costs for SSE in EFGs

However, the practical experiments with reduced costs show that the final sizes of the RGs often reach the size of the original game.

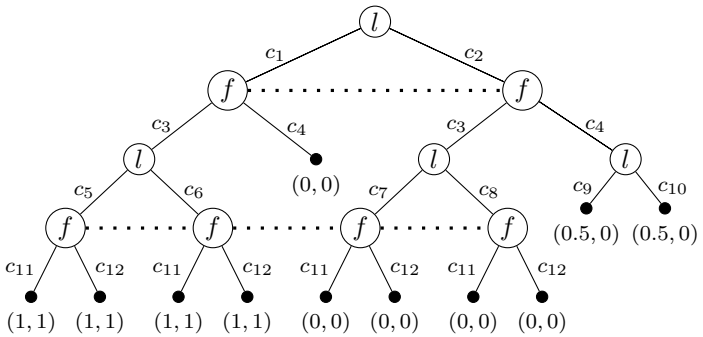



Fig. 2. An example of an EFG where the final RG for the strategy generation with reduced costs is equal to the complete game. The nodes which belong to the same information set are connected by a dashed line. Otherwise, the figure follows a standard denotation of an extensive-form game.

*Example 3.1.* As an example, consider an EFG depicted in Figure 2. At the beginning, the RG is initialized as

$$\Sigma' \leftarrow \{\emptyset, c_1, c_1c_5, c_3, c_4, c_{11}, c_{12}\}. \tag{15}$$

Note that $\Sigma'$ already describes a SEFCE in this game. After the first iteration, the sequences $\hat{\Sigma} = \{c_1c_6, c_2, c_2c_9, c_2c_{10}\}$ are added into the RG. $c_1c_6, c_2c_9$ and $c_2c_{10}$ are included, because they lead to the leafs with the positive utility and no dual variable is able to induce a non-positive reduced cost. The sequence $c_2$ is added because of the positive value of the dual variable $(4)(inf(c_2), \emptyset)$ used in computing $rc(p(c_2, \emptyset))$. The RG is hence extended into

$$\begin{aligned}\Sigma' \leftarrow \{&\emptyset, c_1, c_1c_5, c_1c_6, c_2, c_2c_9, c_2c_{10},\\ &c_3, c_3c_{11}, c_3c_{12}, c_4, c_{11}, c_{12}\}.\end{aligned} \tag{16}$$

In the second iteration, the positive value of the dual variable $(4)(inf(c_2c_7), c_4)$ enforces adding also sequences $c_2c_7, c_2c_8$ and $\Sigma = \Sigma'$, i.e. the RG in equal to the complete game.

The algorithm based on reduced costs expands the whole game tree, even though the SSE is clearly located in the left subtree. Our experiments gave comparable results even when the algorithm was modified to add only the sequences with the highest costs, or by initializing the restricted game using either a leaf with the highest utility for the leader, or a Nash equilibrium in the zero-sum variant of the games where the follower's utilities are set to be complementary to the leader's utilities.

**ALGORITHM 2:** Gadget-Based Algorithm for computing SEFCE.

**Input:** An EFG $G$
**Output:** leader's expected utility and strategy profile in SEFCE
$(M, \Sigma') \leftarrow$ expand(root($G$), $\emptyset$)
$expansionNeeded \leftarrow$ size($M$) $> 0$
**while** *expansionNeeded* **do**
  **for** *state h in M* **do**
    $\Sigma' \leftarrow \Sigma' \cup$ createGadget($h$)
  $(value, p) \leftarrow$ solve($\Sigma'$)
  $M \leftarrow \emptyset$, $E \leftarrow$ getGadgetsToExpand(p)
  $expansionNeeded \leftarrow False$
  **for** *state h in E* **do**
    $(M', \Sigma') \leftarrow$ expand($h, \Sigma'$)
    $M \leftarrow M \cup M'$
    **if** *size*($M'$) $> 0$ **then** $expansionNeeded \leftarrow True$
**return** ($value$, $p$)

The main problem of this approach is in the inability of the reduced costs to evaluate the true impact of a sequence because the utilities are located only in the leafs. In order to design more effective strategy-generation methods, it is necessary to introduce generalized temporary leafs with utilities for non-terminal sequences. When designing the temporary leafs in the Stackelberg setting, we have to keep in mind that the leader can use the individual leafs to create threats against the follower. He can either intentionally lure the follower into a specific subtree or to punish him in another. In contrast to the double-oracle methods, the temporary leafs have to consist of more complex tree structures than just the single game states.

## 4 GADGET-BASED STRATEGY GENERATION FOR EFGS

We introduce a novel strategy-generation algorithm that follows the intuition and avoids adding complete sequences into the game tree by abstracting selected parts of the game tree. We term these abstractions as *gadgets*: a multi-level subtree is replaced with a gadget consisting of a node of the leader and actions leading directly to leafs that correspond to a subset of leafs of the original subtree. The overall structure of the algorithm is the same as in Section 3. In the master problem, we solve for SEFCE in the RG with the gadgets acting as the temporary leafs while ensuring that the gadget-based restricted game always provides an upper bound on SEFCE of the original game. In the slave problem, we identify reachable gadgets and expand the RG. If no gadget is reachable by a current strategy, we reached the equilibrium and it follows that the subtrees rooted in gadgets are not a part of the equilibrium. Algorithm 2 depicts the main steps of the algorithm. We now describe these steps in more details.

***Construction of Gadgets.*** We explain how gadgets are constructed. Let $h$ be the state of the leader which should serve as the root of the gadget. We find all leafs $Z_h$ reachable from $h$. The leafs can be visualized as points in a two-dimensional space where one dimension correspond to the utility of the leader and the second one corresponds to the utility of the follower (see Figure 4 for an example of this visualization). Since these points represent possible outcomes in the abstracted game tree and the goal is to maximize the utility for the leader, it is sufficient to keep only a subset of $Z_h$ that correspond to the upper convex

hull of these points – for each reachable utility of the follower we seek maximal utility of the leader. We add new sequences, such that every gadget leaf is reachable from $h$ by one leader's action. Note that once we add these new sequences to the set of sequences in the RG $\Sigma'$, by the definition of information set, $h$ is no longer a part of the same information set as in the original unrestricted game.

**ALGORITHM 3:** Creating the gadget.

**Input:** A state $h$
**Output:** sequences added to RG
$\hat{\Sigma} \leftarrow \emptyset$
$Z_h \leftarrow$ getLeafsUnder($h$)
$Z_h \leftarrow$ getUpperConvexHull($Z_h$)
**for** *state* $z$ *in* $Z_h$ **do**
  $a_{z,h} \leftarrow$ createNewAction($z$, $h$)
  $\hat{\Sigma} \leftarrow \hat{\Sigma} \cup seq_l(h)a_{z,h}$
**return** $\hat{\Sigma}$

**ALGORITHM 4:** Expanding the gadget.

**Input:** A gadget root $h$, a set of sequences $\Sigma'$
**Output:** new gadget roots, updated set $\Sigma'$
$\Sigma' \leftarrow \Sigma' \backslash \{\sigma \in \Sigma' : \exists z\ \sigma = seq_l(h)a_{z,h}\}$
$H_h \leftarrow$ getShallowestLeaderStates($h$)
**for** *state* $g$ *in* $H_h$ **do**
  $\hat{\Sigma}_l \leftarrow \{\sigma \in \Sigma_l : \sigma \sqsubseteq \sigma_l(g)\}$
  $\hat{\Sigma}_f \leftarrow \{\sigma \in \Sigma_f : \sigma \sqsubseteq \sigma_f(g)\}$
  $\Sigma' \leftarrow \Sigma' \cup \hat{\Sigma}_f \cup \hat{\Sigma}_l$
**return** $(H_h, \Sigma')$

***Expansion of Gadgets.*** Now we describe the expansion of the RG. Let $h$ be the state of the leader which should be expanded and $\Sigma'$ the current set of sequences in the RG. In case $h$ is a root of the gadget, we first delete from $\Sigma'$ all gadget sequences associated with $h$. We search the subtree in the unrestricted game under $h$ and find the set $H_h$ of the shallowest leader's states in each branch, as

$$H_h \leftarrow \{g \in \mathcal{H}_l : \sigma(h) \sqsubseteq seq(g); \nexists g' \in \mathcal{H}_l\ \sigma(h) \sqsubseteq seq(g') \sqsubseteq seq(g)\} \tag{17}$$

For each state $g$ in $H_h$ we add to $\Sigma'$ all prefixes of $seq_l(g)$ and $seq_f(g)$. Finally, all states in $H_h$ are identified as the new gadget roots.

It remains to explain which gadgets have to be expanded, based on the correlation plan $p$ of the current solution. First, we find the set $R$ of all gadgets reachable by $p$ as

$$R \leftarrow \{h \in \mathcal{H}_l : \exists z\ \exists \sigma_f\ p(seq_l(h)a_{z,h}, \sigma_f) > 0\}. \tag{18}$$

We have to ensure the gadgets provide an upper bound on SEFCE. To ensure that, the RG has to be *closed on the information sets of the follower* – whenever a node $h$ where the follower acts is added into the restricted game, all nodes from the information set $I$, such that $h$ belongs to this information set, also have to be added to the restricted game.

LEMMA 4.1. *Let $G'$ be a RG closed on the information sets of the follower. Then for the utility of the leader in SEFCE of $G'$ (denoted as $SEFCE(G')$) it holds that $SEFCE(G') \geq SEFCE(G)$, where $G$ is the complete game.*

PROOF. Because $G'$ is closed on the information sets of the follower, the belief of the follower that he is located in a given state of his information set in RG is fully determined by the strategy of the leader in the RG. Therefore, the strategy of the follower in this information set cannot be affected by any strategy of the leader in the subtrees rooted in the gadgets. Because gadgets assume that leader can always obtain any outcome in the subtree under the gadget, the abstraction represented by a gadget overestimates leader's strategizing ability in the subtree, hence providing an upper bound on $SEFCE(G)$. □

The gadgets which have to be expanded in the current iteration of the algorithm are hence both all the reachable gadgets and also the minimum set of gadgets which will retain the

property of the RG being closed on the information sets, given that all gadgets in $R$ are expanded.

The gadget algorithm creates a restricted game similarly to the strategy-generation with the reduced costs. The RG is again fully defined by the sequences of the leader. However, the set $\Sigma'$ is now updated in a more complex manner, as described in the previous paragraphs about creating the gadgets and expanding the gadgets. Let $\tilde{\Sigma} = \Sigma' \backslash \Sigma$. The sets $\mathcal{H}', \mathcal{Z}', \mathcal{A}'$ and $\mathcal{I}'$ can be derived as follows:

$$\mathcal{H}' \leftarrow \{h \in \mathcal{H} : \forall i \in \mathcal{N}\ seq_i(h) \in \Sigma'\} \cup \{z : \exists h\ seq_l(h)a_{z,h} \in \tilde{\Sigma}\} \tag{19}$$

$$\mathcal{Z}' \leftarrow \mathcal{H}' \cap \mathcal{Z} \tag{20}$$

$$\mathcal{A}'(h) \leftarrow \{a \in \mathcal{A}(h) : ha \in \mathcal{H}'\} \cup \{a_{z,h} : \exists z\ seq_l(h)a_{z,h} \in \tilde{\Sigma}\} \quad \forall h \in \mathcal{H}' \tag{21}$$

$$\mathcal{I}'_i \leftarrow \{I_i \in \mathcal{I}_i : \exists h \in \mathcal{H}' \cap I_i; \nexists z\ seq_l(h)a_{z,h} \in \tilde{\Sigma}\} \cup \{I_h : \exists z\ seq_l(h)a_{z,h} \in \tilde{\Sigma}\} \quad \forall i \in \mathcal{N}. \tag{22}$$

The LP for computing SEFCE is altered according to the current set $\Sigma'$.

LEMMA 4.2. *Let $\hat{G}$ be a modified game constructed from the original game $G$ by forbidding some actions of the leader. Then $SEFCE(\hat{G}) \leq SEFCE(G)$.*

PROOF. Because some actions of the leader are forbidden, the set of strategies available to the leader is restricted, while the set of strategies of the follower remains unconstrained. In case the excluded strategies are not a part of the SEFCE support in $G$, then $SEFCE(\hat{G}) = SEFCE(G)$. Otherwise some leafs of SEFCE become unreachable, the commitment advantage of the leader is hence reduced, resulting in a worse strategy for the leader and $SEFCE(\hat{G}) \leq SEFCE(G)$. □

Now we are ready to prove the algorithm converges to SEFCE.

THEOREM 4.3. *Assume a solution $p$ of Algorithm 2. Then $p$ is a correlation plan of a SEFCE in the original unrestricted game. Algorithm 2 terminates in a finite time.*

PROOF. Since every EFG is finite, there is a finite number of gadgets which can be created. Once a gadget is expanded, it can never be created again, because the game tree is expanded top to bottom. Therefore, there is a finite number of LPs to solve. Algorithm 2 hence terminates in a finite time.

In every iteration, Algorithm 2 finds SEFCE in the current RG, such that the utility of the leader is maximal. In the worst case, the restricted game equals the complete game and it cannot be extended any further and the algorithm terminates, returning a SEFCE. Otherwise, there are states in the complete game, which are not a part of the final RG $G'$. Algorithm 2 terminates once no gadget is reachable by $p$ in the current solution. $G'$ can be hence seen as a modified game in which some actions (those not added to the RG) are forbidden. Therefore, by Lemma 4.2 it holds that $SEFCE(G') \leq SEFCE(G)$. Because RG is closed on the information sets of the follower, by Lemma 4.1 $SEFCE(G') \geq SEFCE(G)$. The SEFCE in the RG is hence a SEFCE in the original unrestricted game. □

### 4.1 Heuristic Gadget-Based Strategy Generation for EFGs

The effectiveness of the complete algorithm described in the previous section is still limited. However, at the cost of losing theoretical guarantees, the algorithm can be turned into a heuristic one and its performance can be measured experimentally. To this end, we introduce three main heuristics: (1) we relax the requirement that the restricted game (RG) has to be

closed on the information sets of the follower; (2) the upper convex hull of points in a gadget can be approximated using fewer points; (3) the leader utility can be artificially decreased in order to compensate for the overestimation caused by the fact that the maximum-value leafs are rarely reached if the follower plays the best response. We applied all these heuristics and we now describe each of the heuristics in more detail:

***Restricted Game Not Closed on Information Sets of the Follower.*** In order to guarantee the convergence to the equilibrium, the requirement for the RG to be closed on follower's information sets is necessary. Consider a variant of our algorithm, where we relax the condition for maintaining the closed RG on the information sets of the follower and we expand only those gadgets that are reachable in the current solution. The following example shows a game where such a modification leads to a suboptimal solution.

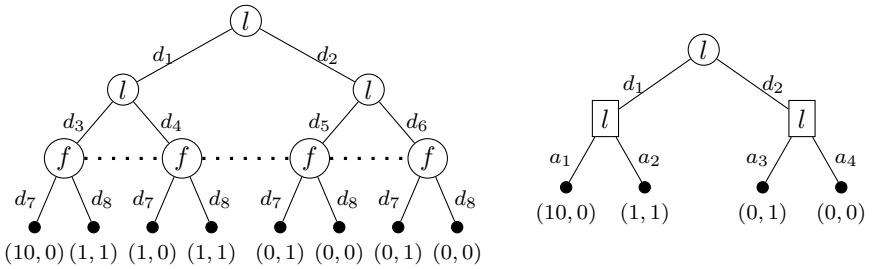



Fig. 3. (Left) An example of EFG where the heuristic algorithms does not converge. The figure follows a standard denotation of an extensive-form game. (Right) An initial RG of the gadget algorithm. By rectangles are denoted the roots of the gadgets. The full gadget algorithm would expand both gadgets after the first iteration. The heuristic gadget algorithm expands only the left gadget.

*Example 4.4.* Consider the game depicted in Figure 3, where SEFCE (which is the same as the SSE) is as follows: the leader commits to playing actions $d_1$ and $d_2$ with the same probability 0.5 and actions $d_3$ and $d_5$ in his lower information sets. The follower is, therefore, indifferent between $d_7$ and $d_8$ and by the definition of SEFCE plays $d_7$. The follower's strategy guarantees the leader the expected utility of 5.0 in the equilibrium. The right part of Figure 3 shows the initial RG of the heuristic gadget algorithm with gadget actions $a_1, a_2, a_3$ and $a_4$. Because the optimal strategy of the leader is to move into the left gadget, it is expanded into the full subtree. In the second iteration, the leader has no intention of playing $d_2$ neither, because his utility is strictly greater in the left subtree. The algorithm therefore terminates. Because the RG is not closed on the follower's information sets, the leader is unaware of the possibility to make the follower indifferent and the optimum is not reached.

***Approximation of Upper Convex Hulls.*** This approximation addresses the problem of creating gadgets for subtrees with large upper convex hull of leafs. Assume that there are $m$ sequences of the follower and $n$ sequences of the leader in the subtree. The number of leafs is hence upper bounded by $m \times n$. If the number of leafs added to the gadget is close to this bound and $mn > m + n$, the number of the sequences which are added into the RG is even larger then number of sequences in the whole subtree and the corresponding LP can be prohibitively large. We solve this issue by excluding from the upper convex hull those leafs, which do not provide a significant increase in the leader's utility when compared with

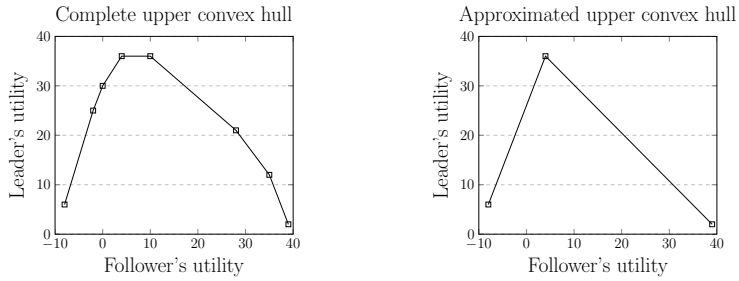



Fig. 4. An example of upper convex hull approximation. (Left) The complete upper convex hull, which is used in the full gadget algorithm. (Right) The approximated upper convex hull with parameter $\delta = 0.4$.

the affine combination of the neighboring leafs. Formally, we delete a leaf $z = (u_l, u_f)$ in between the leafs $z^1 = (u_l^1, u_f^1)$ and $z^2 = (u_l^2, u_f^2)$ in case that

$$\left| u_l^1 + \frac{(u_l^1 - u_l^2)(u_f - u_f^1)}{u_f^1 - u_f^2} - u_l \right| < \delta u_l, \tag{23}$$

where the greater the $\delta > 0$ gets, the more strict the approximation is.

***Penalization of Leader's Utilities.*** Finally, we penalize the leader's utility in order to compensate the optimism when the upper convex hull is created from the leafs directly. Also, during the iterations, there might be several SEFCE with the same optimum, but with different support. In case the solution returned from the LP solver uses the gadgets even though the same optimum can be reached with sequences leading to the ordinary leafs in the RG, the algorithm makes unnecessary iterations. Consequently, both the size of the final RG and the computational time are unnecessarily increased. To speed up the convergence of the algorithm we subtract $\epsilon|maxUtility_l|$, where $\epsilon > 0$, from all leader's utilities in leafs in the gadgets in order to enforce the LP solver into not using the gadgets if not necessary.

## 5 EXPERIMENTS

We performed experiments to evaluate the performance of our heuristic algorithm using strategy generation to compute SSE in EFGs. For comparison, we use the state-of-the-art SEFCE-based algorithm [19], which we refer to as FULL since it uses the full strategy space. All algorithms were implemented in Java 1.8 and all LP computations were completed by a single-threaded IBM CPLEX 12.8 solver using the barrier method. Based on an initial exploration of the parameter space we set the parameter $\delta$ that controls the approximation of the upper convex hulls to value $\delta = 0.3$ since it gave the best results. However, we note that the difference in performance between different values of $\delta$ was relatively small.

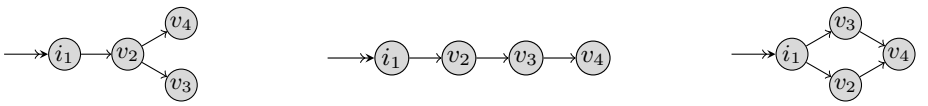
***Flip It Games.*** The domain we used for the experiments is the "Flip It" game [17]. This game is motivated by a cybersecurity scenario where an attacker can perform a stealthy attack to gain control of a resource (e.g., install malware on a host or steal a password) that may not be immediately detected by the defender. However, the defender can take actions to restore control to the defender (e.g., performing a virus scan or resetting a password). Flip It can be viewed as a general model of players competing over resources, and many variations have been proposed in the literature. We use it for evaluation here because it has a symmetric structure that makes it particularly difficult to compute SSE; specifically, the vast majority of follower strategies are best responses to some strategy for the leader. This

makes it a particularly challenging test for our approach since strategy generation methods would perform even better in games with many irrelevant strategies.

A two-player Flip It game is defined as a tuple $F = (V, E, t, \rho, \gamma)$. The game is played by a defender and an attacker on a directed graph $(V, E)$ for a finite number of simultaneous rounds $t$. The graph represents a typical structure for a computer network where not all nodes are publicly accessible and an attacker may need to move deeper into the network. There is a positive reward $\rho : V \to \mathbb{R}^+$ and a positive cost $\gamma : V \to \mathbb{R}^+$associated with each node $v \in V$. At the beginning of the game, the defender controls all of the nodes. In each round, each player selects one node to flip, i.e to attempt to gain control of. The flipping action is successful when two conditions are met. First, the current owner of the node does not also flip it; and second, the player has control over at least one predecessor of the node. We assume that the source nodes in $(V, E)$ can be flipped in any round (i.e., they are public nodes). For every flipping action, the players pay the cost assigned to the node. At the end of every round the players collect the total rewards from all nodes they control:

$$u_j^i(V_j^i) \leftarrow -\gamma(v_j^i) + \sum_{v \in V_j^i} \rho(v) \quad \forall i \in \mathcal{N}, \tag{24}$$

where $V_j^i$ are the nodes the player $i$ owns after round $j$. After $t$ rounds the game ends and the final utilities are the sum of the rewards collected in the individual rounds. We consider two versions of the game with different amounts of information provided to the players. In the All-Points (AP) version the players learn whether their action succeeded and how many points they have in total after each round. In the No-Info (NI) version the only information the players observe is the sequence of actions they play. Moreover, we assume that the graph can also contain a single disconnected *p*ass node with zero reward and zero cost, simulating a pass action. The directed graphs used in the experiments are depicted in Figure 5. All of them include the pass node, which is not depicted in the figure.




(a) Graph 4a (b) Graph 4b (c) Graph 4c

Fig. 5. Graphs used in the experiments with the Flip It game.

We assume the defender acts as a leader in this game, while the attacker assumes the role of the follower. For graphs 4a and 4c in Figure 5 we solved 40 instances of the Flip It games: 20 All-Points games and 20 No-Info games. For graphs 4b we computed 60 instances: 30 All-Points games and 30 No-Info games. Each node in the graph was randomly assigned to one of the following types: (1) high reward, high cost, (2) high reward, low cost, (3) low reward, high cost, and (4) low reward, low cost. The reward and cost were generated uniformly randomly from the intervals depicted in Table 6 to generate representative instances of Flip It games.

The number of rounds was fixed at $t = 5$, so the number of nodes in the EFG representation of the Flip It game is approximately $11 \times 10^6$. Since the SI-LP variant of FULL algorithm was reported to be fastest in [19], we use this variant as a baseline algorithm for computing SSE in Flip It games.

| Interval\Type | (1) | (2) | (3) | (4) |
|---|---|---|---|---|
| Reward interval | 6..10 | 6..10 | 3..6 | 1..4 |
| Cost interval | 6..10 | 3..6 | 5..9 | 1..4 |

Fig. 6. The intervals from which the rewards and costs were generated for individual types of nodes.

### 5.1 Results

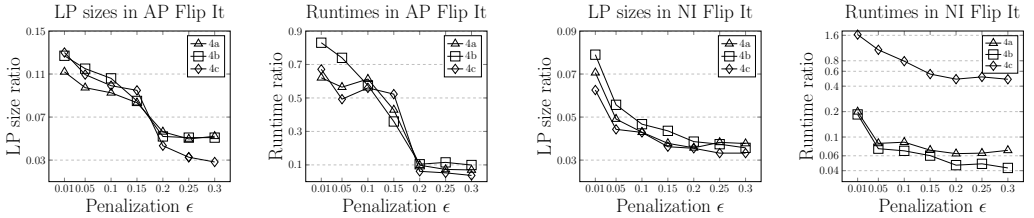



Fig. 7. The median LP sizes and median runtimes in both variants of Flip It for different values of $\epsilon$.

The sizes of the LPs generated for the final restricted games (RG) in the All-Points version of the Flip It games are presented in the leftmost graph of Figure 7. The x-axis shows the penalization parameter $\epsilon$, while the y-axis depicts the ratio of the number of coefficients in the LP describing SSE in the final RG to the number of coefficients in the complete LP used in FULL. Every point in the graph corresponds to the median ratio over the sampled instances. The results show that even with small $\epsilon$, the final LPs are more than eight times smaller than in FULL. As the $\epsilon$ is set higher, the final RGs become even smaller. For the graph 4*c* and $\epsilon = 0.3$, the final LP reaches a size of less than 3% of the full LP. The runtime results for the All-Points version are depicted in the second graph of Figure 7. The x-axis varies the penalization parameter and the y-axis shows the median speedup, which we calculate as the runtime of the heuristic gadget algorithm divided by the runtime of the FULL algorithm.

In the same Figure 7, we present also the results for the final LP sizes and runtimes on the No-Info version of the Flip It game. The graphs follow the same format as the graphs for the All-Points version. Interestingly, when given no information about the progression of the game, the more complex structure of the graph 4*c* slows down the convergence of the heuristic algorithm. The reason is that with larger information sets than in the All-Points version, the upper convex hulls are also larger and contain similar leafs. The algorithm expands only a few gadgets in every iteration and the number of LP invocations is greater. Note that even though the speedup is not that impressive, the sizes of final LPs of all graphs are still less than 4% for the largest value of the parameter $\epsilon$.

Finally, in Table 8 we present the median deviations from the optimum solution achieved by the gadget algorithm. The deviations are measured as the absolute difference in the game values computed by FULL and the heuristic gadget algorithm divided by the SSE game value. We see in the results that the heuristic gadget algorithm achieves nearly optimal results even with higher values of the penalization parameter $\epsilon$. For example, for the graph 4*c* the deviation is 6.38%, with the size of the final LP only 2.8% in the All-Points version, and 1.75% with the size of the final LP 3.3% in the No-Info version.

| Instance \ $\epsilon$ | 0.01 | 0.05 | 0.1 | 0.15 | 0.2 | 0.25 | 0.3 |
|---|---|---|---|---|---|---|---|
| 4a All-Points | 0% | 0% | 0% | 0.9% | 2.42% | 3.01% | 3.23% |
| 4a No-Info | 0% | 0.35% | 0.72% | 1.29% | 1.77% | 2.5% | 2.55% |
| 4b All-Points | 0% | 0% | 0% | 0.56% | 0.8% | 2.39% | 2.48% |
| 4b No-Info | 0% | 0.16% | 0.67% | 1.27% | 2.15% | 2.42% | 2.86% |
| 4c All-Points | 0% | 0% | 0.033% | 0.79% | 3.47% | 4.8% | 6.38% |
| 4c No-Info | 0% | 0.24% | 0.89% | 1.75% | 1.75% | 1.75% | 1.75% |

Fig. 8. The deviations achieved with the heuristic gadget algorithm under different setting of the penalization parameter $\epsilon$.

## 6 CONCLUSION

In this paper, we introduce the first algorithm that successfully exploits the ideas of incremental strategy generation for computing Stackelberg equilibria in extensive-form games. Our algorithm is based on a novel way of representing abstracted parts of the game tree, as well methods for expanding the representation systematically to compute Stackelberg equilibrium. An advantage of our representation is that it leads to three different heuristic approaches that trade off theoretical guarantees for greatly improved practical performance. The heuristic methods can compute exact (or near-optimal) outcomes in practice while constructing only 3% of the original linear program compared to the current state-of-the-art algorithm, often achieving significant speed-up in computation times even in a very challenging class of games.

The current experimental results show that even in a game that is structurally difficult for strategy generation and the Stackelberg solution concept, it is not necessary to consider the whole game tree and all the possibilities. Our algorithm suggests additional directions towards more scalable and approximate algorithms for computing (approximate) Stackelberg equilibrium in large extensive-form games, and it would be interesting to explore other types of games that may have more favorable structure.

## REFERENCES


[1] Branislav Bošanský, Simina Brânzei, Kristoffer Arnsfelt Hansen, Troels Bjerre Lund, and Peter Bro Miltersen. 2017. Computation of Stackelberg Equilibria of Finite Sequential Games. *ACM Transactions on Economics and Computation* 5, 4, Article 23 (Dec. 2017), 24 pages.

[2] Branislav Bošanský, Albert Xin Jiang, Milind Tambe, and Christopher Kiekintveld. 2015. Combining Compact Representation and Incremental Generation in Large Games with Sequential Strategies. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*.

[3] Branislav Bošanský, Christopher Kiekintveld, Viliam Lisý, and Michal Pěchouček. 2014. An Exact Double-Oracle Algorithm for Zero-Sum Extensive-Form Games with Imperfect Information. *Journal of Artificial Intelligence Research* 51 (2014), 829–866.

[4] Branislav Bošanský and Jiří Čermák. 2015. Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. 805–811.

[5] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2015. Heads-up limit holdem poker is solved. *Science* 347, 6218 (2015), 145–149.

[6] Karel Durkota, Viliam Lisý, Christopher Kiekintveld, Branislav Bošanský, and Michal Pěchouček. 2016. Case Studies of Network Defense with Attack Graph Games. *IEEE Intelligent Systems* 31, 5 (2016), 24–30.

[7] Karel Durkota, Viliam Lisý, Christopher Kiekintveld, Karel Horák, Branislav Bošanský, and Tomáš Pevný. 2017. Optimal Strategies for Detecting Data Exfiltration by Internal and External Attackers. In *GameSec 2017, LNCS 10575*. 171–192.

[8] Manish Jain, Erim Kardes, Christopher Kiekintveld, Fernando Ordónez, and Milind Tambe. 2010. Security Games with Arbitrary Schedules: A Branch and Price Approach.. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*.

[9] Manish Jain, Milind Tambe, and Christopher Kiekintveld. 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.

[10] Daphne Koller, Nimrod Megiddo, and Bernhard von Stengel. 1996. Efficient Computation of Equilibria for Extensive two-person Games. *Games and Economic Behavior* (1996), 247–259.

[11] Christian Kroer, Gabriele Farina, and Tuomas Sandholm. 2018. Robust Stackelberg Equilibria in Extensive-Form Games and Extension to Limited Lookahead. In *Proceedings of Thirty-Second AAAI Conference on Artificial Intelligence*.

[12] George Leitmann. 1978. On generalized Stackelberg strategies. *Journal of Optimization Theory and Applications* 26, 4 (1978), 637–643.

[13] Joshua Letchford and Vincent Conitzer. 2010. Computing Optimal Strategies to Commit to in Extensive-Form Games. In *Proceedings of the 11th ACM conference on Electronic commerce*. 83–92.

[14] H. Brendan McMahan, Geoffrey J. Gordon, and Avrim Blum. 2003. Planning in the Presence of Cost Functions Controlled by an Adversary. In *Proceedings of the International Conference on Machine Learning*. 536–543.

[15] Matěj Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. 2017. DeepStack: Expert-Level Artificial Intelligence in No-Limit Poker. *Science* (2017).

[16] Tuomas Sandholm. 2015. Steering evolution strategically: computational game theory and opponent exploitation for treatment planning, drug design, and synthetic biology. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI Press, 4057–4061.

[17] Marten Van Dijk, Ari Juels, Alina Oprea, and Ronald L Rivest. 2013. FlipIt: The game of stealthy takeover. *Journal of Cryptology* 26, 4 (2013), 655–713.

[18] Ondřej Vaněk. 2013. *Computational Methods for Transportation Security*. Ph.D. Dissertation. Czech Technical University in Prague.

[19] Jiří Čermák, Branislav Bošanský, Karel Durkota, Viliam Lisý, and Christopher Kiekintveld. 2016. Using Correlated Strategies for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of Thirtieth AAAI Conference on Artificial Intelligence*. 439–445.

[20] Bernhard von Stengel. 1996. Efficient Computation of Behavior Strategies. *Games and Economic Behavior* 14 (1996), 220–246.

[21] Bernhard von Stengel and Françoise Forges. 2008. Extensive-Form Correlated Equilibrium: Definition and Computational Complexity. *Mathematics of Operations Research* 33, 4 (2008), 1002–1022.

[22] Bernhard von Stengel and Shmuel Zamir. 2004. *Leadership with Commitment to Mixed Strategies*. Technical Report. CDAM Research Report LSE-CDAM-2004-01.

[23] Haifeng Xu, Rupert Freeman, Vincent Conitzer, Shaddin Dughmi, and Milind Tambe. 2016. Signaling in bayesian stackelberg games. In *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 150–158.