Coevolutionary Approach to Sequential Stackelberg Security Games

Adam Żychowski[1] [0000−0003−0026−5183] and
Jacek Mańdziuk[1] [0000−0003−0947−028X]

Faculty of Mathematics and Information Science, Warsaw University of Technology, Koszykowa 75, 00-662 Warsaw, Poland {a.zychowski,j.mandziuk}@mini.pw.edu.pl

Abstract. The paper introduces a novel coevolutionary approach (CoEvoSG) for solving Sequential Stackelberg Security Games. CoEvoSG maintains two competing populations of players' strategies. In the process inspired by biological evolution both populations are developed simultaneously in order to approximate Stackelberg Equilibrium. The comprehensive experimental study based on over 500 test instances of two game types proved CoEvoSG's ability to repetitively find optimal or close to optimal solutions. The main strength of the proposed method is its time scalability which is highly competitive to the state-of-the-art algorithms and allows to calculate bigger and more complicated games than ever before. Due to the generic and knowledge-free design of CoEvoSG, the method can be applied to diverse real-life security scenarios.

Keywords: Coevolution · Security Games · Cybersecurity

1 Introduction

New technologies bring new challenges. One of them is cybersecurity. In recent years, this topic has gained more and more importance [14] since more and more critical systems are connected to the Internet and increasingly many aspects of people's lives depend on reliable computer infrastructure. We are facing a constant arms race between defenders and attackers. One of the approaches to the issue of cybersecurity attacks is to model them as a non-cooperative game. This approach was applied, for instance, in intrusion detection problem in mobile ad-hoc networks [7], security-aware distributed job scheduling in cloud computing [5], detecting vulnerabilities in interbank network [6], planning deep packet inspections [29], and other. In particular, the Stackelberg Security Games (SSGs) recently gained lots of popularity due to a bunch of successful practical applications [19].

SSGs were successfully deployed not only in cybersecurity domain [20, 26] but also in a wide range of real-world scenarios, e.g. scheduling Los Angeles International Airport canine patrols [8], protecting US Coast Guard's resources in Boston harbor [18], or preventing poaching in the Queen Elizabeth National Park in Uganda [4].

In SSG there are two asymmetrical players: the Defender and the Attacker. The Defender commits to their strategy first. Then, the Attacker, knowing the Defender's commitment, decides on their own strategy. The above order of strategy-related decisions favors the Attacker and mimics real-world scenarios in which the Attacker can observe the opponent's strategy (e.g. patrol schedules) and plan their attack accordingly.

The strategy chosen by the Defender is a *mixed one*, i.e. a probability distribution over all possible *pure* (i.e. simple deterministic) strategies [3]. The Attacker is aware of this distribution but has no knowledge about its specific materialization (the sequence of actions that will actually be played). The goal of SSG is to find *Stackelberg Equilibrium* (SE), i.e. the pair of players' strategies that fulfills the following assumption: changing strategy by any player will lead to his/her result deterioration.

In this paper, we consider sequential SSGs which means that each player's strategy consists of a sequence of actions to be executed (played) in consecutive time steps. In such SSGs, finding SE is an NP-hard problem [3]. For this reason, exact methods have limited applicability and are rarely implemented in real-world scenarios. Instead, a number of heuristics approaches were proposed in the literature, including the use of Evolutionary Algorithms (EAs) [12, 28, 29]. EAs are inspired by the process of biological evolution and consists in maintaining a population of potential solutions, which is iteratively modified by applying evolutionary operators: *mutation*, *crossover* and *selection*.

In this paper, we extend the previous EA approaches and propose the coevolutionary algorithm for solving SSGs (CoEvoSG). The method not only maintains a population of Defender's strategies (as EA-based approaches) but also a population of Attacker's strategies. Both populations compete with each other in the process of coevolution. In effect, the convergence to the near-optimal solution is much faster than in the state-of-the-art methods, which allows to solve larger and more complex games than ever before.

Contribution. The contribution of this paper is three-fold: (i) a novel coevolutionary algorithm (CoEvoSG) for Sequential Stackelberg Security Games, capable of finding optimal or near-optimal solutions is proposed, (ii) a comprehensive experimental study proves the efficacy of CoEvoSG and its ability to solve games of sizes and complexity that are beyond the capability of state-of-the-art methods, (iii) to our knowledge, application of coevolutionary algorithms to solving sequential SSGs, has never been considered before in the related literature.

2 Problem definition

A sequential SSG is played by two players: the Defender (D) and the Attacker (A), and is composed of m time steps (moves). In each time step both players simultaneously choose their action to be performed. A *pure strategy* σ_P of player P ($P \in \{D, A\}$) is a list of his/her actions in consecutive time steps: $\sigma_P = (a_1, a_2, \dots, a_m)$. If by Σ_P we denote a set of all possible pure strategies of P, then a probability distribution $\pi_P \in \Pi_p$ over Σ_P is the *mixed strategy* of P, where

Π_p is set of all his/her mixed strategies. For any pair of strategies (π_D, π_A) the expected payoffs for the players are defined and denoted by $U_D(\pi_D, \pi_A)$ and $U_A(\pi_D, \pi_A)$. Stackelberg Equilibrium is a pair of strategies (π_D, π_A) satisfying the following conditions:

$$\pi_D = \underset{\bar{\pi}_D \in \Pi_D}{\arg\max}\, U_A(\bar{\pi}_D, BR(\bar{\pi}_D)), \qquad BR(\pi_D) = \underset{\pi_A \in \Pi_A}{\arg\max}\, U_A(\pi_D, \pi_A).$$

The first equation chooses the best Defender's strategy π_D under the assumption that the Attacker always selects the best response strategy ($BR(\pi_D)$) to the Defender's committed strategy.

Furthermore, if there exists more than one optimal Attacker's response (with the same highest Attacker's payoff), the Attacker selects the one with the highest corresponding Defender's payoff, i.e. breaks ties in favor of the Defender. While this assumption may seem counterintuitive, the opposite way of breaking ties may lead to situations when equilibrium doesn't exist [24]. The above SE extension is known as Strong Stackelberg Equilibrium [1] and is adopted in this paper (as well as in the vast majority of SSG publications).

Both players choose their strategy at the beginning of the game (first the Defender and then the Attacker) and they cannot change it during the gameplay, i.e. in consecutive steps they follow actions encoded in the selected strategy irrespective of the opponent's moves (they are not aware of opponent's current and past actions). Conitzer et al. [3] proved that for each Defender's mixed strategy there exists at least one Attacker's pure strategy which maximizes their payoff. This property is commonly utilized by solutions proposed in the literature since it narrows the Attacker's response search space to only pure strategies.

3 Related work

The methods of solving SSGs can be divided into two main groups: exact and approximate. Exact approaches are based on Mixed-Integer Linear Programming (MILP), which formulates SSG as an optimization problem with a specific target function and a set of linear integer constraints that must be fulfilled. MILP programs are usually computed by specially optimized software engines - *solvers*.
C2016. One of the most popular exact method is C2016 [23], which also bases on MILP but instead of directly computing SE, utilizes the Stackelberg Extensive-Form Correlated Equilibrium (SEFCE). In SEFCE, the Defender can send signals to the Attacker who has to follow them in their choice of strategy. C2016 uses a linear program for computing SEFCE and then modifies it by iteratively restricting the signals the Defender can send to the Attacker and converging to SE. In this article C2016 was used to calculate the reference optimal solutions.
O2UCT. Thee above-mentioned MILP approaches returns exact (optimal) solutions but suffer from exponential computation time and poor memory scalability, which makes them inefficient for large games. Thus, some approximate approaches have been recently proposed, e.g. O2UCT [10, 11] which utilizes an Upper Confidence Bounds applied to Trees (UCT) algorithm [13] (a variant of

Monte Carlo Tree search [21]). O2UCT is based on guided sampling of the Attacker's strategy space and optimizes the Defender's strategy under the assumption that the sampled Attacker's strategy is the optimal response. O2UCT scales visibly better than exact MILP-based solutions and returns close-to-optimal solutions for various types of games.

EASG. Another heuristic method, which is the most related to the approach presented in the paper, is Evolutionary Algorithm for Stackelberg Games (EASG) [27, 28], which optimizes the Defender's payoff by evolving a population of candidate strategies. EASG starts off with a population that contains randomly selected pure Defender's strategies. Then, until the stop condition is not fulfilled, the population evolves in consecutive generations. In each generation, the following four operations are applied: crossover, mutation, evaluation, and selection.

Crossover combines two individuals randomly selected from a population by merging their pure strategies and halving their probabilities. Afterwards, the resultant chromosome is shortened (simplified) by deleting some of its pure strategies with a chance inversely proportional to their probabilities. The mutation operator changes one of the pure strategies encoded in the chromosome starting from a randomly selected time step. New actions are drawn from all feasible actions in a corresponding game state. The role of mutation is to boost exploration of the strategy space.

Next, each individual is assigned a fitness value which is the expected Defender's payoff. This step requires finding the optimal Attacker's response to the mixed Defender's strategy encoded in the chromosome. To this end, EASG iterates over all possible Attacker's pure strategies and selects the one with the highest Attacker's payoff. Due to the potentially large space of Attacker's pure strategies, the evaluation phase is the most time-consuming step of EASG.

Finally, in the selection phase, individuals with higher Defender's payoff are more likely to be selected to the next generation. The above evolutionary approach was successfully applied to various types of SSGs including games with moving targets [12] or games assuming Attacker's bounded rationality [29].

4 Coevolutionary approach

Motivation. As we mentioned in the previous section, EASG evaluation process requires iterating over all possible Attacker's pure strategies in order to find the best one and calculate the expected Defender's payoff. This evaluation procedure is performed thousands of times (for each individual in each generation) which is infeasible (too time-consuming), except for small games.

Furthermore, in many SSG instances there exists a relatively small subset of Attacker's strategies that need actually to be considered when looking for the optimal response. Many of the Attacker's strategies can either be trivially qualified as weak (e.g. an attack at a well-protected target with low reward or a sequence of actions which does not lead to a target), or there are subsets of similar strategies and only one representative from each of them needs to be examined in order to find the best Attacker's response. However, for more complex games it is not possible to determine - within a reasonable time - which of the Attacker's strategies could be omitted, nor to define a representative subset of these strategies, due to the high dependence of this selection on the game topology (structure) and payoff distribution.

In order to address the issue of time-consuming evaluation process in EASG, we propose a novel coevolutionary approach which maintains two populations: one composed of the Defender's mixed strategies (as in EASG) and the other consisting of the Attacker's pure strategies. Strategies from the Attacker's population are used to evaluate the Defender's strategies. Instead of calculating the Defender's payoff against all possible Attacker's pure strategies, it is now calculated only versus a subset of the Attacker's strategies represented in the population. Both populations compete with each other, i.e. the Attacker's population attempts to find the best possible response to the strategies from the Defender's population and vice versa - the Defender's population tries to evolve the most effective strategies with respect to the response strategies encoded in the Attacker's population.

System overview. A general overview of the CoEvoSG algorithm is presented in Figure 1. Both populations are initialized with random pure strategies and then developed alternately. First, the Attacker's population is modified by evolutionary operators (crossover, mutation, and selection) through g_p generations. Then, the Defender's population is evolved through the same number of g_p generations. The above loop is repeated until the stop condition is not satisfied.

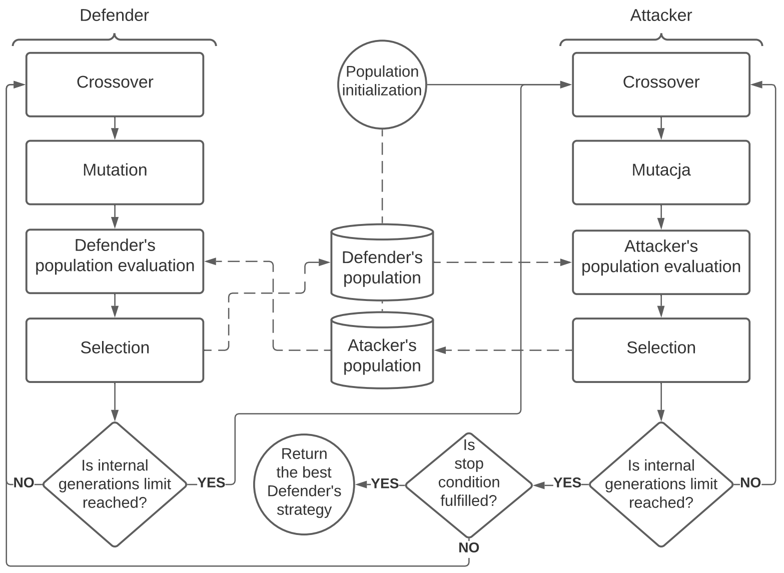

Defender
Attacker
Crossover
Mutation
Defender's population evaluation
Selection
Population initialization
Defender's population
Atacker's population
Crossover
Mutacja
Attacker's population evaluation
Selection
Is internal generations limit reached?
NO
YES
Return the best Defender's strategy
Is stop condition fulfilled?
YES
NO
Is internal generations limit reached?
YES
NO

Fig. 1: A high-level overview of the CoEvoSG algorithm.

All evolutionary operators applied to the Defender's population are implemented in the same way as in EASG and briefly described in Section 3. Additional details can be found in [28]. The novel operators applied to the Attacker's population are described below.

Initialization. The Attacker's population contains N_A individuals. Each individual k represents a randomly selected pure Attacker's strategy, encoded as a list of actions in consecutive time steps: $\sigma_A^k = (a_1^k, a_2^k, \ldots, a_m^k)$. In each time step $t \in \{1, \ldots, m\}$ a_t^k is drawn uniformly from all feasible actions in a given state.

Crossover. Each individual from the Attacker's population is selected for a crossover with probability p_c. Selected individuals are paired randomly and for each pair, one-point crossover is performed, i.e. for strategies $\sigma_A^r = (a_1^r, a_2^r, \ldots, a_m^r)$ and $\sigma_A^s = (a_1^s, a_2^s, \ldots, a_m^s)$ the following two child individuals are created: $\sigma'^r_A = (a_1^r, \ldots a_i^r, a_{i+1}^s, \ldots, a_m^s)$ and $\sigma'^s_A = (a_1^s, \ldots a_i^s, a_{i+1}^r, \ldots, a_m^r)$, where $a_i^r = a_i^s$ is the first common action (in the same time step) in the parent strategies. If such an action does not exist, the crossover has no effect. For example, if an action is to choose a vertex in a game graph the player moves to, then $a_i^r = a_i^s$ would be the first common vertex on the paths defined by the parent strategies.

Mutation. Each individual is mutated with probability p_m. Mutation operator, starting from a randomly selected step, modifies all subsequent actions encoded in the chromosome. Each subsequent action is chosen randomly from all available actions in the current state. The result of mutation of strategy $\sigma_A^r = (a_1^r, a_2^r, \ldots a_m^r)$ is $\sigma'^r_A = (a_1^r, a_2^r, \ldots, a_{i-1}^r, a_i^{r'}, a_{i+1}^{r'}, \ldots, a_m^{r'})$, where i is the chosen time step. The role of mutation is to boost exploration of new areas in the search space by means of an introduction of random perturbations.

Evaluation. The evaluation procedure is the most important component of the proposed solution. Individuals from the Defender's population are evaluated against all strategies from the Attacker's population. For each Defender's strategy (π_D) the outcome (players' payoffs) of the gameplays against all strategies from the Attacker's population are computed. Then, the best Attacker's response is chosen: $\sigma_A^{best} = \arg\max_{\sigma_A} U_A(\pi_D, \sigma_A)$. Finally, the expected Defender's payoff against this Attacker's response ($U_D(\pi_D, \sigma_A^{best})$ is assigned as the fitness value of the evaluated Defender's strategy π_D. There is a chance that the above fitness value is not the true expected Defender's payoff because of the lack of the (overall) optimal Attacker's response in the Attacker's population. However, the expected algorithm's behavior is to evolve such a strategy (optimal response) in the coevolution process in subsequent generations.

The evaluation procedure of the individuals from the Attacker's population is more complicated. Usually, there is no single optimal Attacker's response for all Defender's strategies. Depending on the particular Defender's commitment (Defender's mixed strategy), the best Attacker's response may change.

It is generally desired that the Attacker's population is composed of optimal responses for all possible Defender's strategies. Assigning the average Attacker's payoff against all strategies from the Defender's population (or part of it) as fitness value may be a weak approach because a given Attacker's strategy is usually strong (optimal) only against specific Defender's strategies. Such Attacker's strategy needs to be preserved but averaging the payoffs will decrease the fitness of such a strategy posing a risk of omitting it in the selection process.

Hence, a better idea is to use the maximum metric. However, in Defender's population (in order to preserve its diversity) there exist also some weaker strategies. For those strategies most of the Attacker's strategies will lead to high Attacker's payoff and such an approach wouldn't allow distinguishing good Attacker's strategies from the bad ones (because all of them will get high fitness as a maximum payoff against one of the weak Defender's strategies). This observation discredits calculating maximum payoff against all Defender's strategies. On the other extreme, the Attacker's fitness value could be computed only against the best strategy from the Defender's population, but this would lead to degeneration (premature convergence) of the Attacker's population. All Attacker's strategies would tend to be an optimal response for a particular Defender's strategy, becoming vulnerable to other strategies from the Defender's population.

Consequently, an intermediate option was implemented, i.e. the Attacker's strategy fitness is the maximum of Attacker's payoffs against the N_{top} highest fitted individuals from the Defender's population (N_{top} is CoEvoSG parameter).
Selection. The selection process decides which individuals from the current population will be promoted to the next generation. At the beginning, e individuals with the highest fitness value are unconditionally transferred to the next generation. They are called *elite* and preserve the best-fitted solutions. Then, a *binary tournament* is repeatedly executed until the next generation population is filled with N_A individuals. For each tournament, two individuals are sampled (with replacement) from the current population (including those affected by crossover and/or mutation). The higher fitted chromosome wins (and is promoted to the next generation) with probability p_s (so-called selection pressure parameter). Otherwise, the lower-fitted one is promoted.
Stop condition. The algorithm ends when at least one of the following conditions is satisfied: (a) CoEvoSG attained the maximum number of l_g generations, (b) no improvement of the best-found solution (Defender's payoff) was observed in consecutive l_c generations. Only generations referring to the Defender's population are considered when verifying the above conditions.

5 Experimental setup

5.1 Benchmark games

CoEvoSG was tested on two popular SSG benchmarks: FlipIt and Warehouse Games, previously used for testing state-of-the-art methods, e.g. in [11, 17, 23].

FlipIt Games. FlipIt Games (FIGs) [22] reflect certain cybersecurity scenarios. The Attacker attempts to gain control over some elements of network infrastructure (e.g. computers, routers, mobile devices) and the Defender can take actions to regain control of the infected units.

FIGs are played on directed graphs with n vertices, for a fixed number of m time steps. In each time step, players simultaneously select one vertex which they want to take control of (to *flip* the node). At the beginning, only some subset of vertices (entry nodes) is available for the Attacker. This mimics the scenario in which some part of the network infrastructure is publicly accessible from the outside (e.g. Internet). The Attacker starts penetrating the network from one of those entry nodes. Taking control over the vertex (flip action) is successful only if the two following conditions are fulfilled: (1) the player controls at least one of predecessor vertices (unless it is an entry node), (2) the current owner of this vertex does not take the flip action on it in the same time step.

At the beginning of the game, all vertices are controlled by the Defender. Each node has assigned two values: a reward (> 0) for controlling it, and a cost (< 0) of taking a flip attempt. The final player's payoff is calculated by summing the rewards in all nodes controlled by that player after each time step and the costs of all flip attempts (either successful or not). Figure 2 presents a sample FIG scenario.

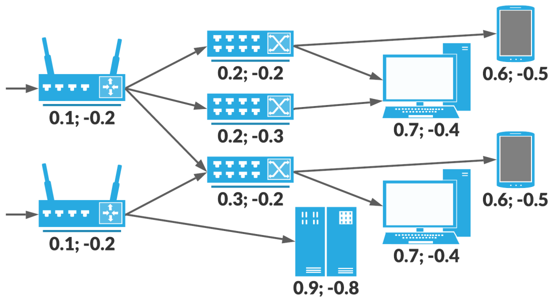


Fig. 2: Example FIG scenario with two entry nodes (routers) on the left. Numbers below each component denote a reward for controlling the node (left) and the cost of a flip attempt (right).

In the experiments, 280 FIG instances were generated randomly with the following parameters: $m \in \{3, 4, 5, 6, 8, 10, 15, 20\}$, $n \in \{5, 10, 15, 20, 25, 30, 40\}$. For each pair (m, n) 5 games were created with random payoffs (rewards drawn from $(0, 1)$, costs from $(-1, 0)$) and random graph structures (generated according to Watts–Strogatz model [25] with an average vertex degree $d_{avg} = 3$).

The experiments were performed in *No-Info* variant [2] which means that the players were not aware of whether their flip action succeeded or failed.

Warehouse Games. Warehouse Games (WHGs) [9] are inspired by real estate (warehouses or residential buildings) protection scenarios. The games are played on undirected graphs with n vertices, for m time steps. A subset of special vertices are called targets (T). Graph edges represent corridors and vertices symbolize rooms. At the beginning, the Defender and the Attacker are placed in the predetermined starting vertices. In each time step, each player's action consists in moving to one of the neighbor vertices (connected with an edge) or staying in the current vertex.

The game ends in one of the following circumstances: (a) both players are located in the same vertex v in the same time step - then, the Attacker is "caught" and the players are given payoffs associated with that vertex: $U^v_{D+} > 0$ (Defender) and $U^v_{A-} < 0$ (Attacker); (b) the Attacker reaches one of the targets $t \in T$ and is not caught (there is no Defender in this target) - in this case, the attack is successful and the players receive payoffs $U^t_{D-} < 0$ (Defender) and $U^t_{A+} > 0$ (Attacker); (c) none of above conditions is satisfied - both players receive a payoff of 0. Figure 3 presents a sample WHG scenario.

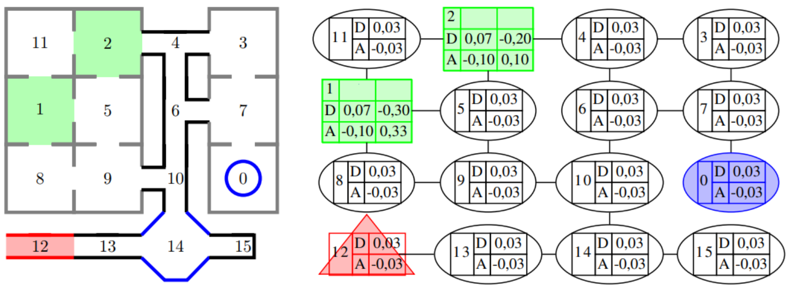


Fig. 3: An example WHG scenario: warehouse layout (left) and the corresponding graph (right) with payoffs of the players in the respective game outcomes. Green rectangular vertices are targets, a red triangle vertex and a blue circle vertex are the Attacker's and the Defender's starting points, respectively

For CoEvoSG evaluation 240 WHG instances were generated with $m \in \{3, 4, 5, 6, 8, 10, 15, 20\}$ and $n \in \{15, 20, 25, 30, 40, 50\}$ (5 games per each (m, n) pair). Players' payoffs were drawn from $[-1; 1]$. The number of targets depended on a graph size: $|T| = \lceil \frac{n}{5} \rceil$. Graphs were generated according to Watts–Strogatz random graphs model [25] with an average vertex degree $d_{avg} = 3$.

5.2 Parameterization

All common EASG and CoEvoSG parameters were set according to the recommendations proposed for EASG [28]. Namely, the Defender's population size $N_D = 200$, crossover probability $p_c = 0.8$, mutation probability $p_m = 0.5$, selection pressure $p_s = 0.9$, elite size $e = 2$, maximal number of generations $l_g = 1000$, maximal number of generations with no improvement $l_c = 20$. The parameters of evolutionary operators (mutation, crossover, selection) for the Attacker's population were assigned the same values as for the Defender's population, however, CoEvoSG requires several new parameters which need to be tuned. In order to find their recommended values, a set of parameter tuning experiments with 50 random games, different from the test WHG instances, were performed. FIGs have similar game structure and do not require separate parametrization.

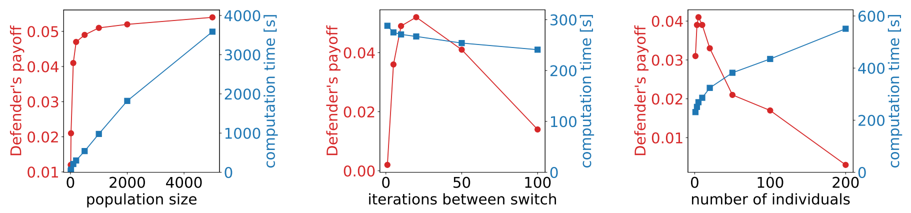
The first tested parameter was the Attacker's population size (N_A). The following values were considered: $\{10, 20, 100, 200, 500, 1000, 2000, 5000\}$. The results (average Defender's payoff and computation time) are presented in Figure 4a. Clearly, the bigger the Attacker's population size the better the results as the Defender's payoff is calculated more accurately. If the Attacker's population contained all possible Attacker's pure strategies, then the Defender's individuals evaluation would be an exact value (not an approximation) since the optimal Attacker's response would always be present in the Attacker's population. However, as stated previously one of the motivations for introducing coevolution is to speed up the Defender's strategies evaluation by checking them only against a representative subset of all Attacker's strategies. Thus, based on the presented results, $N_A = N_D = 200$ was set.

Another tested parameter was the number of consecutive generations for each player - g_p. Please recall that in CoEvoSG Defender's and Attacker's populations are evolved alternately in the batches of g_p generations. The results of tuning this parameter are presented in Figure 4b. Small values ($g_p \leq 5$ - frequent switching between populations), as well as big ones ($g_p \geq 50$) result in performance deterioration. Infrequent switching makes one population dominant - the other one stagnates over a long time with no chances to response to the evolved individuals from the other population. At the same time, for all tested values computation time is similar. Hence, $g_p = 20$ was adopted as a recommended value.

The last tuned parameter was N_{top}, i.e. the number of the best individuals from the Defender's population involved in the Attacker's strategies evaluation. The result for $N_{top} \in \{1, 3, 5, 10, 20, 50, 100, 200\}$ presented in Figure 4c confirm our previous conjecture formulated in in Section 4 about the harmfulness of using the whole Defender's population ($N_{top} = 200$). Also, small values of this parameter ($N_{top} < 5$) lead to weaker results due to the presence of some oscillations within the population. In the extreme case of $N_{top} = 1$ (evaluation of a given Attacker's strategy is based on the best Defender's strategy only), we observed that the Attacker's population quickly losses diversity/degenerates. Individuals in the population become similar to one another because they are optimized with respect to only one Defender's strategy. As a result the Attacker's population returns a good response only to this one specific Defender's strategy, and in the next coevolution phase the Defender's population is able to find with ease another strategy for which there is no good response in the Attacker's population. Afterwards, the whole Attacker's population again adapts to the new best Defender's strategy and "forgets" the previous ones. $N_{top} = 10$ appeared to be the best compromise between these two extremes (Figure 4c).

6 Results and discussion

Payoffs. Tables 1 and 2 present average Defender's payoffs with respect to the number of graph nodes and time steps, resp., for the methods described in Section 3. Dashes mean that a particular algorithm was not able to compute some of the test instances within the limit of 100h per game. The results are averaged over 20 independent runs per game.



(a) Attacker's population size (N_A). (b) Consecutive generations per population (g_p). (c) Individuals to evaluate Attacker's strategy (N_{top}).

Fig. 4: Comparison of the average Defender's payoffs and computation times for CoEvoSG specific parameters: N_A, g_p, N_{top}.

Table 1: Average Defender's payoffs with respect to the number of graph nodes.

FIG				
n	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
5	0.890	0.887	0.886	0.886
10	0.854	0.851	0.847	0.845
15	0.811	0.807	0.802	0.798
20	-	0.784	0.780	0.772
25	-	-	0.754	0.746
30	-	-	-	0.730
40	-	-	-	0.722

WHG				
n	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
15	0.052	0.051	0.051	0.050
20	0.054	0.053	0.052	0.050
25	0.048	0.046	0.045	0.043
30	-	0.044	0.042	0.039
40	-	-	0.040	0.036
50	-	-	-	0.029

Presented outcomes show only minor differences between the evolutionary approach (EASG) and proposed coevolutionary algorithm (CoEvoSG). The average differences equal 0.0032 and 0.0020 for FIG and WHG instances, resp. Please note that EASG is a natural reference point for CoEvoSG since CoEvoSG approximates the Defender's payoff (in the evaluation procedure) while EASG computes it thoroughly. A relatively small difference in Defender's payoffs between the methods is a consequence of frequent existence (in over 84% of the cases) of the optimal Attacker's response in their population. The fitness function in such cases returns the same evaluation in both methods.

O2UCT slightly outperforms EASG and CoEvoSG but the differences are not statistically significant - p-values are 0.34 and 0.12, respectively (according to one-tailed t-test). For 23% of games, CoEvoSG returned better result than O2UCT whereas O2UCT was superior in 39% cases (for the remaining 38% of games the outcomes of both methods were the same).

The exact MILP method (C2016) was able to solve 45 FIG and 60 WHG test instances within the allotted time. For these games, CoEvoSG returned the optimal strategy (a difference in Defender's payoff less than $\varepsilon = 0.0001$) in 29/45 (64%) and 38/60 (68%) cases, resp. The average differences between optimal results and CoEvoSG outcomes equaled 0.0137 (FIGs) and 0.0023 (WHGs).

Overall, CoEvoSG was able to solve much bigger games than any of the competitive methods, while returning only slightly weaker Defender's payoffs (whenever comparable).

Table 2: Average Defender's payoffs with respect to the number of time steps.

FIG					WHG				
m	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**	m	**C2016**	**O2UCT**	**EASG**	**CoEvoSG**
3	0.823	0.821	0.820	0.817	3	0.043	0.043	0.043	0.043
4	0.817	0.812	0.808	0.805	4	0.052	0.050	0.050	0.049
5	0.810	0.801	0.798	0.791	5	0.055	0.054	0.053	0.052
6	-	0.794	0.792	0.791	6	0.058	0.056	0.054	0.051
8	-	0.789	0.784	0.781	8	-	0.053	0.051	0.048
10	-	-	0.780	0.778	10	-	-	0.048	0.044
15	-	-	-	0.774	15	-	-	-	0.040
20	-	-	-	0.761	20	-	-	-	0.038

Computation scalability. Figure 5 illustrates computation time of tested methods with respect to the number of graph nodes and time steps. In all cases, the advantage of CoEvoSG is clear. The method preserves near-constant computation time irrespective of game size, while other methods scale approximately linearly (O2UCT and EASG) or exponentially (C2016). Computational complexity of CoEvoSG is approximately constant with respect to the graph size or the number of steps because the algorithm maintains the Defender's and the Attacker's populations of fixed size, independently of other game parameters.

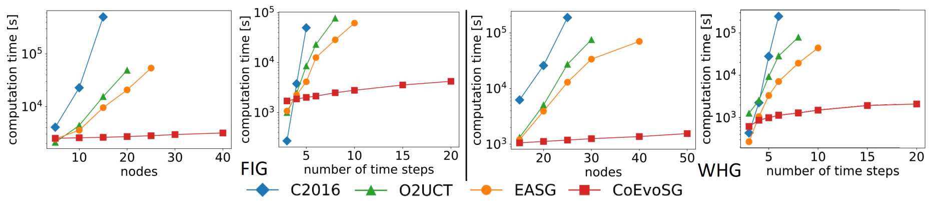


Fig. 5: Comparison of computation time (logarithmic scale) with respect to number of graph nodes and time steps for FIG (left) and WHG (right) games.

In summary, presented results demonstrate that despite slightly worse average Defender's payoffs the proposed coevolutionary approach, thanks to excellent time scalability, offers a viable alternative to both exact and approximate state-of-the-art methods, especially in the case of larger games which are beyond the capacity of the existing algorithms.

7 Conclusions

The paper proposes a novel coevolutionary algorithm for solving sequential Stackelberg Security Games. The method develops two competing populations of players' strategies by specially designed evolutionary operators.

Experimental evaluation performed on two well-established game types with more than 500 test instances have proven the efficacy of the proposed method -

in the majority of test cases optimal solutions were found. The results are is on par with other approximate methods - O2UCT and EASG. However, the true strength of CoEvoSG lies in its time efficiency. It scales visibly better than all state-of-the-art methods and stands out with near-constant computation time irrespective of the game size. Thanks to this property CoEvoSG can be employed to solve arbitrarily large games which are beyond the capacity of the methods proposed hitherto. Moreover, the method is generic and can be easily adapted to other genres of Security Games. What's more, CoEvoSG is an *anytime* algorithm, i.e. is capable of returning a valid solution at any time of the execution process.

CoEvoSG can be directly applied to various real-life cybersecurity problems modelled by FlipIt Games, such as password reset policies, cloud auditing, or supervisory control and data acquisition in industrial internet of things [15].

Our future plans concentrate on extending CoEvoSG to games with multiple Defenders and/or Attackers [16] with the corresponding increase of the number of populations.

Acknowledgement The project was funded by POB Research Centre Cybersecurity and Data Science of Warsaw University of Technology within the Excellence Initiative Program - Research University (ID-UB).

References

1. Breton, M., Alj, A., Haurie, A.: Sequential stackelberg equilibria in two-person games. Journal of Optimization Theory and Applications **59**(1), 71–97 (1988)
2. Černỳ, J., Bošanskỳ, B., Kiekintveld, C.: Incremental strategy generation for Stackelberg equilibria in extensive-form games. In: Proceedings of the 19th ACM Conference on Economics and Computation. pp. 151–168 (2018)
3. Conitzer, V., Sandholm, T.: Computing the optimal strategy to commit to. In: Proceedings of the 7th ACM conference on Electronic commerce. pp. 82–90 (2006)
4. Fang, F., Nguyen, T.H., Pickles, R., Lam, W.Y., Clements, G.R., An, B., Singh, A., Tambe, M., Lemieux, A.: Deploying paws: Field optimization of the protection assistant for wildlife security. In: Proceedings of the 28th Innovative Applications of Artificial Intelligence Conference (2016)
5. Gąsior, J., Seredyński, F.: Security-aware distributed job scheduling in cloud computing systems: a game-theoretic cellular automata-based approach. In: International Conference on Computational Science. pp. 449–462 (2019)
6. Guleva, V.Y.: Estimation of tipping points for critical and transitional regimes in the evolution of complex interbank network. In: International Conference on Computational Science. pp. 432–444 (2020)
7. Guo, Y., Zhang, H., Zhang, L., Fang, L., Li, F.: Incentive mechanism for cooperative intrusion detection: An evolutionary game approach. In: International Conference on Computational Science. pp. 83–97 (2018)
8. Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., Ordónez, F.: Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. Interfaces **40**(4), 267–290 (2010)
9. Karwowski, J., Mańdziuk, J.: A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. European Journal of Operational Research **277**, 255–268 (2019)

10. Karwowski, J., Mańdziuk, J.: Stackelberg Equilibrium Approximation in General-Sum Extensive-Form Games with Double-Oracle Sampling Method. In: Proceedings of the 18th AAMAS conference. pp. 2045–2047 (2019)
11. Karwowski, J., Mańdziuk, J.: Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games. In: Proceedings of the 34th AAAI conference. vol. 34, pp. 2054–2061 (2020)
12. Karwowski, J., Mańdziuk, J., Żychowski, A., Grajek, F., An, B.: A memetic approach for sequential security games on a plane with moving targets. In: Proceedings of the 33rd AAAI Conference. vol. 33, pp. 970–977 (2019)
13. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: European conference on machine learning. pp. 282–293. Springer (2006)
14. Lezzi, M., Lazoi, M., Corallo, A.: Cybersecurity for industry 4.0 in the current literature: A reference framework. Computers in Industry **103**, 97–110 (2018)
15. Liu, Z., Wang, L.: FlipIt Game Model-Based Defense Strategy Against Cyberattacks on SCADA Systems Considering Insider Assistance. IEEE Transactions on Information Forensics and Security **16**, 2791–2804 (2021)
16. Lou, J., Smith, A.M., Vorobeychik, Y.: Multidefender security games. IEEE Intelligent Systems **32**, 50–60 (2017)
17. Oakley, L., Oprea, A.: Qflip: An adaptive reinforcement learning strategy for the flipit security game. In: International Conference on Decision and Game Theory for Security. pp. 364–384. Springer (2019)
18. Shieh, E., An, B., Yang, R., Tambe, M., Baldwin, C., DiRenzo, J., Maule, B., Meyer, G.: Protect: A deployed game theoretic system to protect the ports of the united states. In: Proceedings of the 11th AAMAS conference. pp. 13–20 (2012)
19. Sinha, A., Fang, F., An, B., Kiekintveld, C., Tambe, M.: Stackelberg Security Games: Looking Beyond a Decade of Success. In: Proceedings of the 27th IJCAI conference. pp. 5494–5501 (2018)
20. Sinha, A., Nguyen, T.H., Kar, D., Brown, M., Tambe, M., Jiang, A.X.: From physical security to cybersecurity. Journal of Cybersecurity **1**(1), 19–35 (2015)
21. Świechowski, M., Godlewski, K., Sawicki, B., Mańdziuk, J.: Monte Carlo Tree Search: A Review of Recent Modifications and Applications (2021), https://arxiv.org/abs/2103.04931
22. Van Dijk, M., Juels, A., Oprea, A., Rivest, R.L.: Flipit: The game of "stealthy takeover". Journal of Cryptology **26**(4), 655–713 (2013)
23. Čermák, J., Bošanský, B., Durkota, K., Lisý, V., Kiekintveld, C.: Using correlated strategies for computing stackelberg equilibria in extensive-form games. In: Proceedings of the 30th AAAI Conference. pp. 439–445 (2016)
24. Von Stengel, B., Zamir, S.: Leadership with commitment to mixed strategies. Tech. rep., CDAM Research Report (2004)
25. Watts, D.J., Strogatz, S.H.: Collective dynamics of 'small-world' networks. Nature **393**(6684), 440–442 (1998)
26. Zhang, Y., Malacaria, P.: Bayesian stackelberg games for cyber-security decision support. Decision Support Systems p. 113599 (2021)
27. Żychowski, A., Mańdziuk, J.: A Generic Metaheuristic Approach to Sequential Security Games. In: Proceedings of the 19th AAMAS. p. 2089–2091 (2020)
28. Żychowski, A., Mańdziuk, J.: Evolution of Strategies in Sequential Security Games. In: Proceedings of the 20th AAMAS conference. pp. 1434–1442 (2021)
29. Żychowski, A., Mańdziuk, J.: Learning attacker's bounded rationality model in security games. In: Proceedings of the 28th ICONIP. vol. CCIS 1516, pp. 530–539 (2021)