Journal of Computational Science 68 (2023) 101980

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Coevolution of players strategies in security games

Adam Żychowski [a,*], Jacek Mańdziuk [a,b]

[a] Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland
[b] Institute of Computer Science, AGH University of Science and Technology, Krakow, Poland

ARTICLE INFO

Keywords:
Coevolution
Security games
Cybersecurity
Stackelberg equilibrium

ABSTRACT

Stackelberg Security Games (SSGs) gained recently a lot of attention and popularity due to a bunch of successful practical applications in the field of security maintenance. SSGs model real-life security scenarios as non-cooperative games between the security forces (e.g. secret service, police) and the attackers (e.g. terrorists, military groups). The paper proposes a novel coevolutionary method (CoEvoSG) for solving SSGs that develops two competing populations of player strategies, in the process inspired by biological evolution, so as to approximate the Stackelberg Equilibrium (game solution). CoEvoSG is experimentally evaluated on over 800 test instances of three types of games with various characteristics. The results and their detailed analysis presented in the paper prove the CoEvoSG ability to repetitively find optimal or close to optimal solutions with time scalability excelling the state-of-the-art methods. Consequently, CoEvoSG is capable of calculating solutions for games bigger and more complex than ever before. This study extends our previously published conference paper Żychowski and Mańdziuk (2022).

1. Introduction

One of the main challenges brought by globalization are international security issues, such as terrorism, large-scale thefts, cyberattacks, drugs or weapon smuggling. In recent years, new technologies and scientifically grounded methods have helped to increase the security level and to struggle with the above-mentioned crime activities [1,2]. Among them, Stackelberg Security Games (SSGs) is a rapidly developing research area, related to countering criminal activities, that models tactical security scenarios in the form of a game between security forces and organized attackers (e.g. terrorists, criminals, military groups, etc.).

SSGs gained lots of popularity due to a bunch of successful practical applications [3]. For instance, they were successfully deployed in cybersecurity domain [4,5] and in a wide range of real-world scenarios, e.g. scheduling the Los Angeles International Airport canine patrols [6], protecting Boston Harbor by the US Coast Guards [7] or preventing poaching in the Queen Elizabeth National Park in Uganda [8].

SSG formulation assumes two asymmetrical non-cooperative 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 deduce the opponent's strategy (e.g. by observing patrol schedules) and plan their attack accordingly.

In practice, the Defender chooses a *mixed strategy*, i.e. a probability distribution over all possible *pure* (i.e. simple and deterministic) strategies [9]. The Attacker is aware of this distribution but has no knowledge about its specific materialization (the sequence of actions that will actually be played during the strategy realization). The goal of SSG is to find *Stackelberg Equilibrium* (SE), i.e. a pair of player strategies such that changing the strategy by any of the players would lead to his/her result deterioration.

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. Finding SE is an NP-hard problem [9], and therefore, exact methods have limited applicability. In order to address the scalability problem we have proposed several heuristics approaches relying on Monte Carlo Tree Search [10,11] or Evolutionary Algorithms (EAs) [12–16].

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

* Corresponding author.
E-mail addresses: a.zychowski@mini.pw.edu.pl (A. Żychowski), mandziuk@mini.pw.edu.pl (J. Mańdziuk).

https://doi.org/10.1016/j.jocs.2023.101980
Received 14 October 2022; Received in revised form 25 January 2023; Accepted 23 February 2023
Available online 2 March 2023

1.1. Contribution

The main contribution of this paper is three-fold:

- a novel coevolutionary algorithm (CoEvoSG) for solving Sequential Stackelberg Security Games, capable of finding optimal or near-optimal solutions is proposed,
- a comprehensive experimental study proves the efficacy of CoEvoSG and its ability to solve games of size and complexity that are beyond the capability of the state-of-the-art methods,
- an in-depth analysis of the method's performance and parameterization is presented.

This study extends our previous conference paper [15] mainly in the 3 following dimensions:

- a novel evaluation method of the Attacker's population that improves the results presented in [15] is proposed in Section 6,
- experimental results on a new type of Security Games. substantially different from those considered in [15], are presented in Sections 7 and 5.3,
- a detailed analysis of the method's performance and the interactions between the competing populations is presented in Section 8.

2. Problem definition

A sequential SSG is played by two players: the Defender (D) and the Attacker (A). It 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: $\sigma_P = (a_1, a_2, \dots, a_m)$. Let us denote by Σ_P 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 the set of all his/her mixed strategies. For any pair of strategies (π_D, π_A), the expected payoffs of the players are defined and denoted by $U_D(\pi_D, \pi_A)$ and $U_A(\pi_D, \pi_A)$, respectively. Stackelberg Equilibrium is a pair of strategies (π_D, π_A) satisfying the following conditions:

$$\pi_D = \arg\max_{\bar{\pi}_D \in \Pi_D} U_A(\bar{\pi}_D, BR(\bar{\pi}_D)), \tag{1}$$

$$BR(\pi_D) = \arg\max_{\pi_A \in \Pi_A} U_A(\pi_D, \pi_A) \tag{2}$$

The first equation chooses the best (i.e yielding the highest payoff) 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.

According to *Strong Stackelberg Equilibrium* defined in [17], if in (2) 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 does not exist [18]. The above variant of Stackelberg Equilibrium (i.e. Strong Stackelberg Equilibrium) is considered in this paper, as well as in the 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. This means that in consecutive steps they follow actions encoded in the selected strategy irrespective of the opponent's moves (they are not aware of the opponent's current and past actions).

Lemma 1. *For each Defender's mixed strategy, there exists at least one Attacker's pure strategy which maximizes their payoff.*

The proof of the above lemma can be found in [9]. This property is commonly utilized by the solution methods proposed in the literature since it narrows down the Attacker's response search space to pure strategies only. Likewise, we have built on this Lemma when developing the CoEvoSG algorithm.

3. Related work

3.1. Exact methods

Methods of solving sequential SSGs can be divided into two main groups: exact and approximate. Exact approaches are based on Mixed-Integer Linear Programming (MILP) [19], 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*.

3.1.1. BC2015

The first notable approach from this group was BC2015 [20] - an extension of DOBBS algorithm [19] (which was designed to solve a simpler class of one-step Security Games) to extensive-form games [9]. BC2015 transforms an extensive-form game into its equivalent sequence-form representation which reduces the size of the linear program from exponential (as in DOBBS) to linear with respect to the game tree size.

3.1.2. C2016

Another popular exact method is C2016 [21]. It 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 the experimental evaluation presented in [21], it has been proven that C2016 is more time-efficient than BC2015, therefore we will apply C2016 to calculate the reference optimal solutions.

3.2. Heuristic approaches

The above-mentioned MILP approaches return exact (optimal) solutions but suffer from exponential computation time and poor memory scalability, which makes them inefficient for large games. Thus, a group of approximate approaches have been recently proposed

3.2.1. O2UCT

One of them is O2UCT [10,11] which utilizes an Upper Confidence Bounds applied to Trees (UCT) algorithm [22], a variant of Monte Carlo Tree Search method [23].

O2UCT consists in a guided sampling of the Attacker's strategy space and optimizing the Defender's strategy under the assumption that the sampled Attacker's strategy is the optimal response. The method scales visibly better than exact MILP-based solutions and returns close-to-optimal solutions for variable types of games.

3.2.2. EASG

Another heuristic approach, Evolutionary Algorithm for Security Games (EASG) [13,24], bases on Evolutionary Algorithms that are inspired by the process of biological evolution. EASG maintains a population of potential solutions by iteratively applying evolutionary operators: *mutation*, *crossover* and *selection*. The method optimizes the Defender's payoff by evolving a population of candidate Defender's strategies, i.e. each chromosome represents a valid Defender's mixed strategy. EASG starts off with a population of chromosomes, each of them containing one, randomly selected *pure* Defender's strategy with assigned probability of occurrence equal to 1. 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 [13]. Mutation changes one of the pure strategies encoded in the chromosome starting from a randomly selected time step. New actions are drawn from the set of all feasible actions in a corresponding game state.

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 (cf. Lemma 1) and selects the one with the highest Attacker's payoff. Due to the potentially large space of the 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. Selection is performed by means of a binary tournament which repeatedly selects two chromosomes and promotes the one with the higher fitness value with a given probability $p_s > 0.5$ and the lower-fitted one with probability $1-p_s$. Moreover, a few chromosomes with the highest fitness function values are unconditionally copied to the next generation in order to preserve the best solutions found so far. The above evolutionary approach was successfully applied to various types of SSGs, including games with moving targets [12], anti-poaching games [16] (the so-called Green SSGs), or games which assume that the Attacker is not perfectly rational (i.e. his/her decisions can deviate from the optimal ones in a certain way) [14].

4. Coevolutionary approach

4.1. Motivation

As we mentioned in the previous section, EASG evaluation process requires iterating over all possible Attacker's pure strategies (according to Lemma 1) in order to find the best response strategy (2) and then calculate the expected Defender's payoff. This evaluation procedure is performed thousands of times (for each individual in each generation) which may be time-infeasible, for larger games. In the extreme case, the Attacker's strategy space may be continuous (with infinitely many strategies) which would render the EASG approach [13] ineffective.

Furthermore, in many SSG instances there exists a relatively small subset of Attacker's strategies that actually need 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 subset needs to be examined in order to find the best Attacker's response. However, for more complex games it is generally difficult to *a priori* (or within a reasonable time) recognize such irrelevant strategies or create a generic procedure that would return the representative subset of the Attacker's strategies, due to their high dependence on a game topology (structure) and payoff distribution.

In order to address the above issue, we propose a novel coevolutionary approach which maintains two populations: one composed of the Defender's mixed strategies (as in EASG) and the other one 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 Attacker's 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. The Attacker's population is also subject to the evaluation process (like the Defender's population in EASG).

4.2. System overview

A general overview of the CoEvoSG algorithm is presented in Fig. 1. CoEvoSG maintains two populations: the first one contains the encoded Defender's mixed strategies and the other one consists of the Attacker's pure strategies.

Populations are developed alternately, i.e. 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 satisfied.

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 [13]. The novel operators applied to the Attacker's population are described below.

4.3. 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: $\sigma_A^k = (a_1^k, a_2^k, \dots, a_m^k)$. In each time step $t \in \{1, \dots, m\}$ a_t^k is drawn uniformly from all feasible actions in a given state.

4.4. Crossover

The Attacker's population contains pure strategies so the approach similar to crossover in Defender's population from EASG (which contains mixed strategies) cannot be applied. Each individual from the Attacker's population is selected for crossover with probability p_c. Selected individuals are paired randomly and for each pair one-point crossover is performed, i.e. for strategies $\sigma_A^r = (a_1^r, a_2^r, \dots, a_m^r)$ and $\sigma_A^s = (a_1^s, a_2^s, \dots, a_m^s)$ the following two child individuals are created: $\sigma_A'^r = (a_1^r, \dots, a_i^r, a_{i+1}^s, \dots, a_m^s)$ and $\sigma_A'^s = (a_1^s, \dots, a_i^s, a_{i+1}^r, \dots, a_m^r)$, 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.

The frequency of situations in which crossover has no effect due to non-existence of a common action in the same time step in crossed chromosomes strongly depends on a game characteristics. For bigger games (with high numbers of possible actions in each state) crossover may have no effect in the majority of the cases. For this reason, the crossover probability (p_c) is set close to 1.

Please note that crossover combines strategies of two individuals by swapping parts of their actions and the result of this operation contains only existing actions (either from the first or the second parent). Thus, the crossover is unable to create strategies with new (not yet chosen) actions. The introduction of new actions into the chromosomes is left to the mutation operation (described below), the main role of which is to boost the exploration of new areas in the search space.

4.5. Mutation

Each individual is mutated with probability p_m. The mutation operator, starting from a randomly selected step, modifies all subsequent actions encoded in a chromosome. Each subsequent action is chosen randomly from all available actions in the current state. The result of mutation of strategy $\sigma_A^r = (a_1^r, a_2^r, \dots, a_m^r)$ is $\sigma'^r_A = (a_1^r, a_2^r, \dots, a_{i-1}^r, a_i^{r'}, a_{i+1}^{r'}, \dots, a_m^{r'})$, where i is the chosen time step.

4.6. Evaluation

The evaluation procedure is a crucial component of the proposed solution. Individuals from the Defender's population are evaluated against

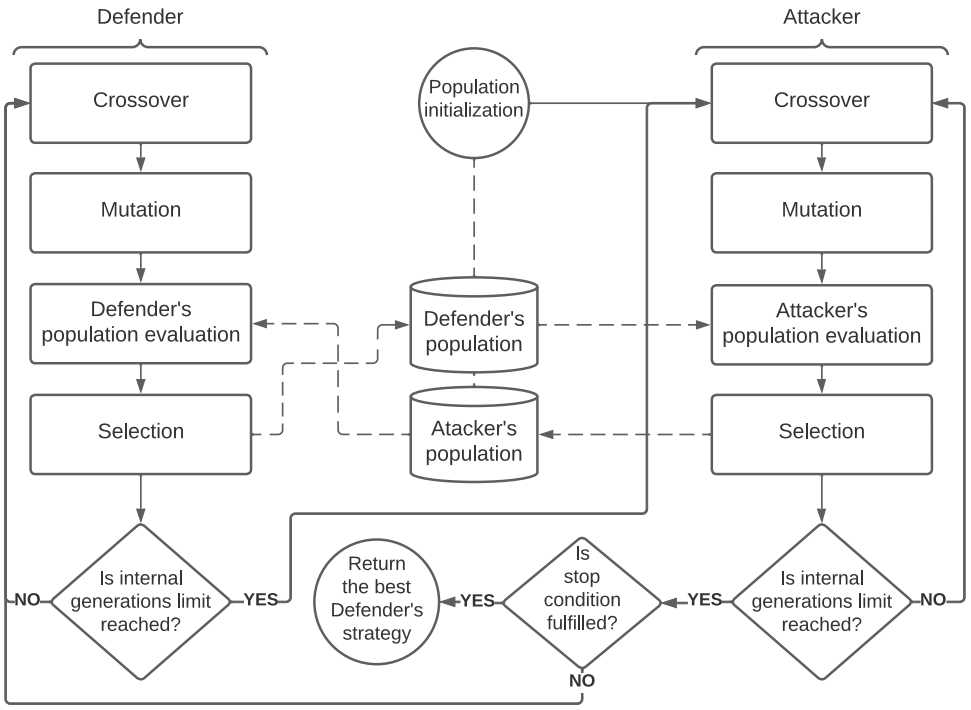


Fig. 1. A high-level overview of the CoEvoSG algorithm.

all strategies from the Attacker's population. For each Defender's strategy (π_D) the outcome (player 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 to 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 of individuals from the Attacker's population is more complicated. Usually, there is no single optimal Attacker's response to all Defender's strategies. Depending on a particular Defender's commitment (Defender's mixed strategy), the best Attacker's response may change. Thus, the Attacker's strategy fitness is the maximum of Attacker's payoffs against the N_{top} highest-fitted individuals from the Defender's population and N_{random} random ones. For further discussion and justification of this evaluation procedure please see Section 8.3. The value of N_{top} is established experimentally in the parameters tuning process, described in Section 6.

4.7. 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 so far 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, the so-called selection pressure parameter. Otherwise, the lower-fitted chromosome is promoted.

4.8. Stop condition

The algorithm ends when at least one of the following conditions is satisfied:

(i) CoEvoSG attained the maximum assumed number of generations – l_g,
(ii) 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 development are considered when verifying the above conditions.

5. Benchmark games

We have tested CoEvoSG on three popular SSG benchmarks: FlipIt Games, Warehouse Games, and Search Games. All of them are frequently used for testing state-of-the-art methods, e.g. in [11,13,21,25].

5.1. Flipit games

FlipIt Games (FIGs) [26] are inspired by cybersecurity scenarios, in which the Attacker attempts to gain control over some elements of the 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 (*flip* this node). At the beginning of the game, all vertices are controlled by the Defender and only a subset of them (entry nodes) is accessible to 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 following two conditions are fulfilled: (1) the player controls at least one of the

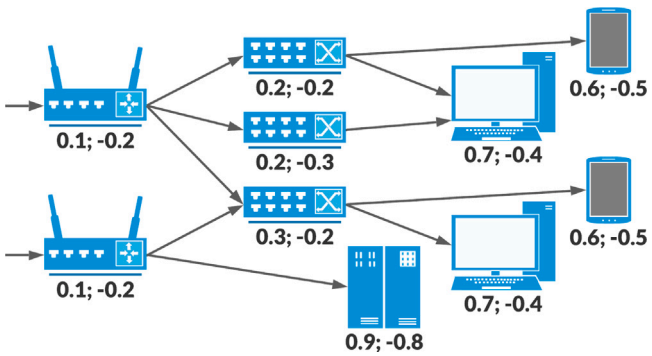


Fig. 2. Example FIG scenario with two entry nodes (routers) on the left. Numbers below each component denote a reward for controlling the node (left) and a cost of a flip attempt (right).

predecessor vertices (unless it is an entry node), and (2) the current owner of this vertex does not take the flip action on this vertex at the same time step.

Each node has assigned two values: a reward (> 0) for controlling this node, 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 across all time steps and the costs of all flip attempts (either successful or not) during the entire game. Fig. 2 presents a sample FIG scenario.

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 [27] with an average vertex degree $d_{avg} = 3$.

The experiments were performed in *No-Info* variant [28] which means that the players were not aware if their flip action was successful. Thus, their strategies were independent of the opponent's actions.

5.2. Warehouse games

Warehouse Games (WHGs) [29] are inspired by the 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, 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 neighboring vertices (connected with an edge) or staying in the current vertex.

The game ends in one of the following situations:

(a) both players are located in the same vertex v at the same time step. This means that 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). This means that the attack is successful and the players receive payoffs $U^t_{D-} < 0$ (Defender) and $U^t_{A+} > 0$ (Attacker),

(c) none of the above conditions are met, in which case both players receive a payoff of 0.

Fig. 3 presents a sample WHG scenario. All WHG instances used in the experiments can be found in [30].

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). Player payoffs were drawn from $[-1; 1]$ interval. The number of targets depended on a graph size: $|T| = \left\lceil \frac{n}{5} \right\rceil$. Graphs were generated according to Watts–Strogatz random graphs model [27] with an average vertex degree $d_{avg} = 3$.

5.3. Search games

Search Games (SEG), introduced in [20], are played on directed graphs. The Attacker's goal is to reach one of the distinguished target vertices, starting from a fixed initial vertex. Contrary to WHG, in SEG the Defender has several units and, furthermore, the movement of each of them is restricted to a specific subset of vertices the unit is allowed to visit.

Another crucial difference compared to WHG is the property of partial observability. Namely, the Attacker leaves traces in visited vertices which can be discovered by a Defender's unit if it visits the node after the Attacker's presence (in one of the subsequent time steps). However, the Attacker has the ability to erase such a trace if they spend an additional time step in a given vertex (i.e. stay in this vertex in two or more consecutive time steps). The end-of-game conditions are the same as in WHG: either the Defender obtains a positive payoff for catching the Attacker, or the Attacker is rewarded for reaching a target vertex without being intercepted, or the game ends with neutral payoffs after a certain number of time steps.

Leaving/discovering traces and possessing more than one unit by the Defender clearly distinguish SEG from the previously described WHG model. Fig. 4 presents an example Search Game with 5 targets, 3 Defender's units and 32 nodes.

In the evaluation process 300 SEG instances were generated with $m \in \{3, 4, 5, 6, 8, 10, 15\}$ and $n \in [15, 50]$. The number of targets $|T|$ varied from 2 to 6.

Since SEGs introduce partial observability (by means of leaving traces) the Defender's pure strategy is no longer in the form of a simple list of actions/moves in consecutive time steps. It has to be extended by the Defender's reaction to discovering the Attacker's traces. This extension is described in detail in [13] and also adopted in this paper.

6. Parameterization

All common EASG and CoEvoSG parameters were set according to the recommendations proposed for EASG [13]. 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 $l_c = 20$. Parameters of the evolutionary operators (mutation, crossover, selection) in the Attacker's population were assigned the same values as in the Defender's population. However, CoEvoSG requires several new parameters which need to be tuned. In order to find their recommended values, a number of parameter tuning experiments on 50 random WHG games, different from the WHG test instances, were performed. In order to verify the robustness of CoEvoSG, there were no additional parameterization on either the FIG or SEG game instances. All of those games were generated independently of the benchmark set (described in the previous subsection) used for CoEvoSG evaluation.

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 Fig. 5(a). 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,

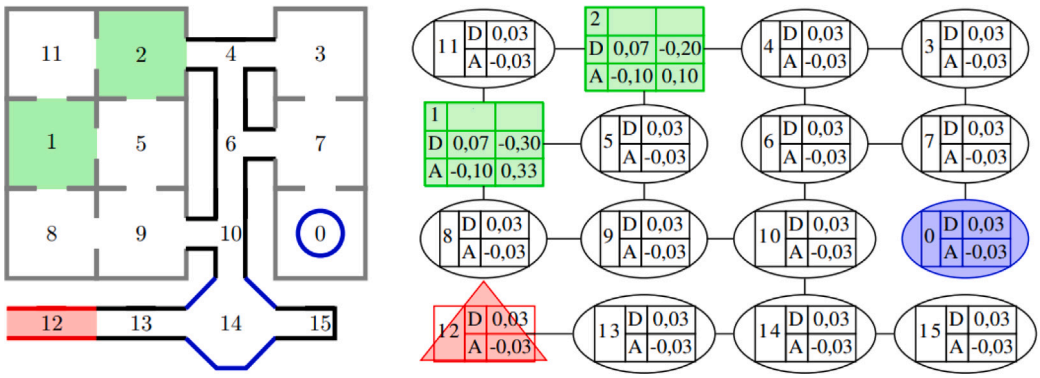


Fig. 3. Example WHG scenario: warehouse layout (left) and the corresponding graph (right) with payoffs of the players related to the respective game outcomes. Green rectangular vertices are targets, a red triangle vertex and a blue circle vertex are the Attacker's and the Defender's starting point, respectively.

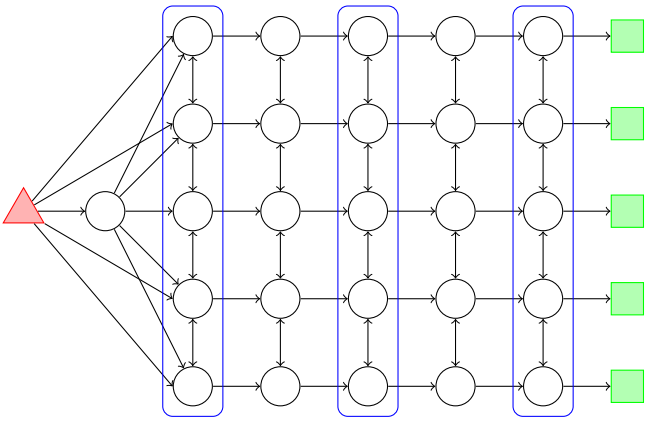


Fig. 4. Example of Search Game topology. A red triangle denotes the Attacker's starting vertex, green rectangles are targets. Three rounded groups of vertices represent restricted subsets of nodes within which each of three Defender's units can freely move.

based on the presented results, N_A was set to 200 which equals the Defender's population size (N_D).

Another tested parameter was the number of consecutive generations g_p for each player. Recall that in CoEvoSG the Defender's and the Attacker's populations are evolved alternately in the batches of g_p generations. The results of tuning g_p are presented in Fig. 5(b). 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 respond to the evolved individuals from the other population. At the same time, for all tested values the 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 Fig. 5(c) confirm our previous conjecture formulated 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 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. 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 particular 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 (Fig. 5(c)).

However, calculating the Attacker's payoff only for the N_{top} Defender's strategies with the highest payoff may not be the best approach when they are very similar to each other. Thus, adding some other (weaker) strategies may be beneficial. To verify this hypothesis experiments with and without adding random individuals were performed. Namely, in the first scenario each individual from the Attacker's population was evaluated against N_{top} best Defender's strategies and in the second scenario apart from those strategies also N_{random} random Defender's strategies were added to the evaluation process. Experimental results show that indeed adding those random supplementary strategies increases the final Defender's payoff returned by CoEvoSG.

The best results were obtained for $N_{random} = 10$ and $N_{top} = 10$. Fig. 6 presents a comparison of the average Defender's payoff with respect to N_{top} with and without addition of random strategies. Incorporation of a bunch of random Defender's strategies into the Attacker's evaluation process extends the initial CoEvoSG formulation [15] and further improves the results.

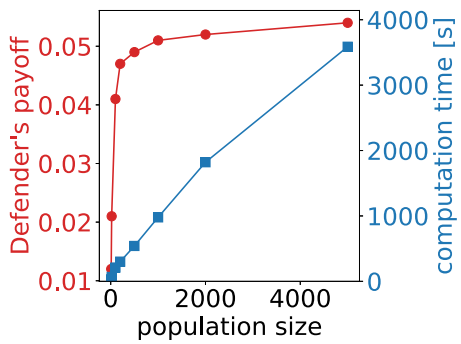
7. Results

7.1. Payoffs

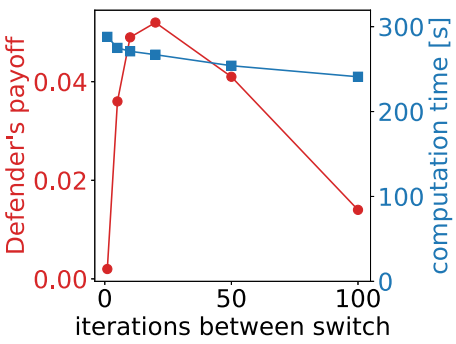
Tables 1, 2 and 3 present the average Defender's payoffs with respect to the number of graph nodes and time steps, for FIG, WHG and SEG games, respectively. The results are also compared with the initial version of the algorithm [15] that does not consider randomly selected Defender's strategies during the evaluation of the Attacker's population, denoted by CoEvoSG⋆. Dashes mean that a particular algorithm was not able to compute some of the test game instances within the limit of 100 h per instance. The results are averaged over 20 independent runs per game instance.

Presented outcomes show only slight deterioration of results when comparing the evolutionary approach (EASG) with the proposed coevolutionary algorithm (CoEvoSG). The average differences are equal to 0.0030, 0.0018 and 0.0024 for FIG, WHG and SEG instances, respectively. Please note that EASG is a natural baseline for CoEvoSG since CoEvoSG approximates the Defender's payoff (in the evaluation procedure) while EASG computes it directly. A relatively small difference in Defender's payoffs between the methods stems from the frequent existence (in over 84% of the cases) of the optimal Attacker's response in CoEvoSG population. In such cases the fitness function returns the same evaluation for both methods.

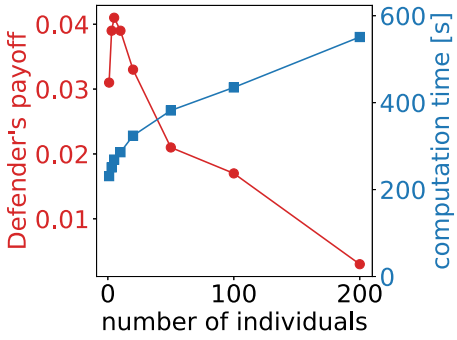
A comparison with CoEvoSG⋆ (a version of CoEvoSG presented in [15]) shows that adding random individuals from the Defender's population during the Attacker strategies' evaluation process leads to



(a) Attacker's population size (N_A).



(b) Consecutive generations per population (g_p).



(c) Number of best individuals used to evaluate the Attacker's strategy (N_{top}).

Fig. 5. Comparison of the average Defender's payoffs and computation times for CoEvoSG parameters used in the evaluation of the Attacker's population.

Table 1
Average Defender's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for FlipIt Games.

n	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
5	0.890	0.887	0.886	0.886	0.886
10	0.854	0.851	0.847	0.845	0.846
15	0.811	0.807	0.802	0.798	0.800
20	–	0.784	0.780	0.772	0.775
25	–	–	0.754	0.746	0.748
30	–	–	–	0.730	0.731
40	–	–	–	0.722	0.724
m	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
3	0.823	0.821	0.820	0.817	0.818
4	0.817	0.812	0.808	0.805	0.806
5	0.810	0.801	0.798	0.791	0.794
6	–	0.794	0.792	0.791	0.791
8	–	0.789	0.784	0.781	0.782
10	–	–	0.780	0.778	0.779
15	–	–	–	0.774	0.776
20	–	–	–	0.761	0.763

statistically significantly better outcomes. For all 3 game types, we obtained p-value < 0.05 according to 1-tailed paired t-test.

O2UCT slightly outperforms EASG and CoEvoSG but the differences are not statistically significant - p-values are equal to 0.34, 0.27 and 0.12, respectively for FIG, WHG and SEG, according to one-tailed t-test. For 21% of the games, CoEvoSG returned better results than O2UCT, whereas O2UCT was superior in 41% of the cases (for the remaining 38% of the games the outcomes of both methods were equal).

The exact MILP method (C2016) was able to solve 45 FIG, 60 WHG and 60 SEG 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%), 38/60 (68%) and 17/60 (28%) cases, respectively. The average differences between the optimal results and CoEvoSG outcomes equaled 0.0137 (FIG), 0.0023 (WHG) and 0.0102 (SEG).

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

7.2. Computation scalability

Figs. 7 and 8 illustrate the computation time of the tested methods with respect to the number of graph nodes and time steps. In all cases, the advantage of CoEvoSG is clear. The method maintains near-constant computation time irrespective of the game size, while other methods scale approximately linearly (O2UCT and EASG) or exponentially (C2016). The approximately constant time complexity of CoEvoSG is caused by maintaining fixed-size Defender's and Attacker's populations, regardless of the remaining game parameters.

Table 2
Average Defender's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for Warehouse Games.

n	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
15	0.052	0.051	0.051	0.050	0.050
20	0.054	0.053	0.052	0.050	0.051
25	0.048	0.046	0.045	0.043	0.044
30	–	0.044	0.042	0.039	0.040
40	–	–	0.040	0.036	0.038
50	–	–	–	0.029	0.031
m	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
3	0.043	0.043	0.043	0.043	0.043
4	0.052	0.050	0.050	0.049	0.049
5	0.055	0.054	0.053	0.052	0.052
6	0.058	0.056	0.054	0.051	0.052
8	–	0.053	0.051	0.048	0.049
10	–	–	0.048	0.044	0.046
15	–	–	–	0.040	0.041
20	–	–	–	0.038	0.040

Table 3
Average Defender's payoffs with respect to the number of graph nodes (top) and time steps (bottom) for Search Games.

n	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
15	0.122	0.116	0.115	0.115	0.115
20	0.117	0.112	0.106	0.101	0.104
25	–	0.123	0.117	0.115	0.116
30	–	–	0.136	0.135	0.135
40	–	–	–	0.150	0.152
50	–	–	–	0.139	0.144
m	C2016	O2UCT	EASG	CoEvoSG⋆	CoEvoSG
3	0.137	0.126	0.118	0.117	0.118
4	0.124	0.113	0.110	0.107	0.109
5	0.106	0.093	0.090	0.085	0.087
6	–	0.129	0.123	0.122	0.123
8	–	–	0.112	0.110	0.111
10	–	–	–	0.142	0.144
15	–	–	–	0.151	0.154

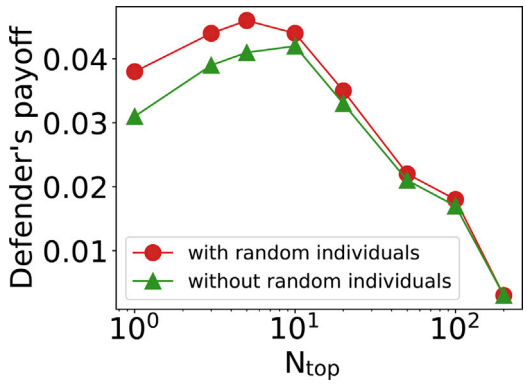


Fig. 6. Comparison of two versions of the Attacker's population evaluation: with and without using random individuals from the Defender's population.

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.

8. Performance analysis and discussion

The results presented in the previous section demonstrate the robustness of CoEvoSG and its ability to find close to optimal strategies with significantly lower computation time. In this section, a more detailed analysis of CoEvoSG performance is presented, e.g. how both populations interact with each other, and how they evolve in time and maintain diversity.

8.1. Interactions between populations

Fig. 9 presents an example CoEvoSG run in terms of maximum Defender's and Attacker's payoffs calculated in the algorithm's evaluation procedure. It can be observed that at the beginning the Defender's population quickly improves. The estimated Defender's payoff is increasing, although, this does not mean that the real Defender's payoff (when calculated against the overall optimal Attacker's response) gets higher, because the Attacker's population is weak and does not yet contain adequate response strategies. However, during the evolution process, the Attacker's population becomes stronger and sometimes finds the optimal response (cf. generation 70 in Fig. 9). Starting from generation 145, until the end of the process the Attacker's population consists of strong and robust Attacker's strategies and the Defender payoff's estimation becomes accurate.

8.2. Structure and diversity of both populations

It may be interesting to investigate not only interactions between the best player strategies but also between the entire populations. To this end, we generated a matrix that presents the Defender's payoff for each pair of strategies from the Defender's and the Attacker's populations (see Fig. 10). First of all, it can be noticed that there is no single Attacker's response that is optimal for all Defender's strategies. Some of the Attacker's strategies are good for certain subsets of the Defender's strategies whereas weak for the others. Populations are clearly diverse but there are also individuals very similar to each other in both the Defender's and the Attacker's populations — they can be recognized as rectangles of uniform color.

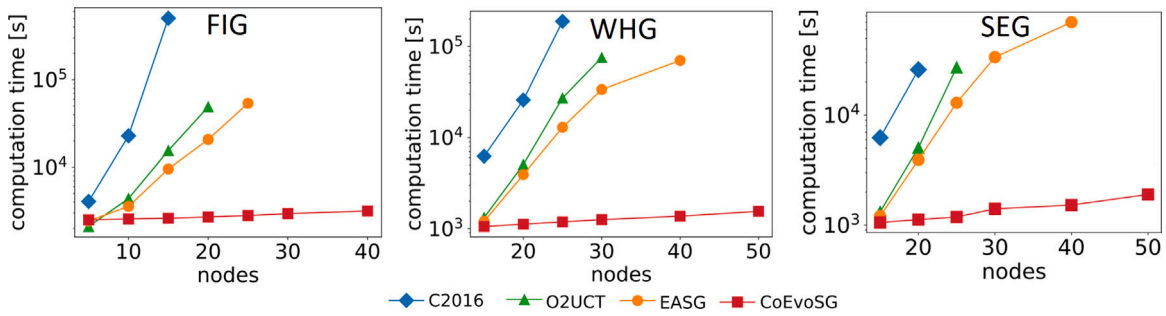


Fig. 7. Comparison of computation time (logarithmic scale) with respect to the number of graph nodes.

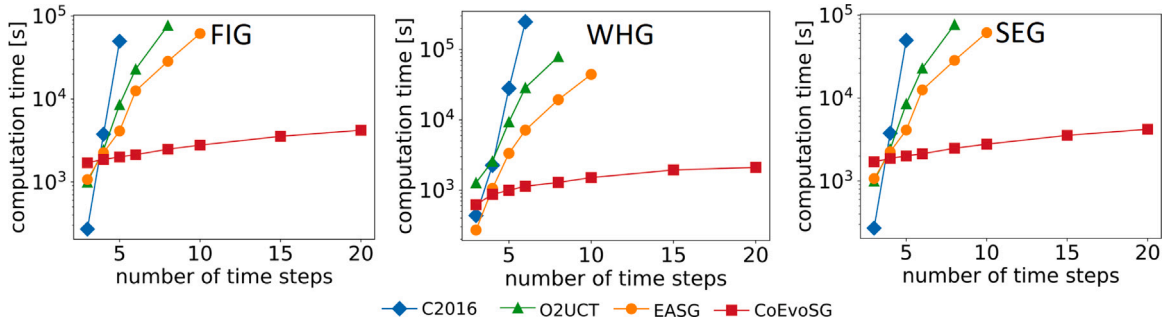


Fig. 8. Comparison of computation time (logarithmic scale) with respect to the number of time steps.

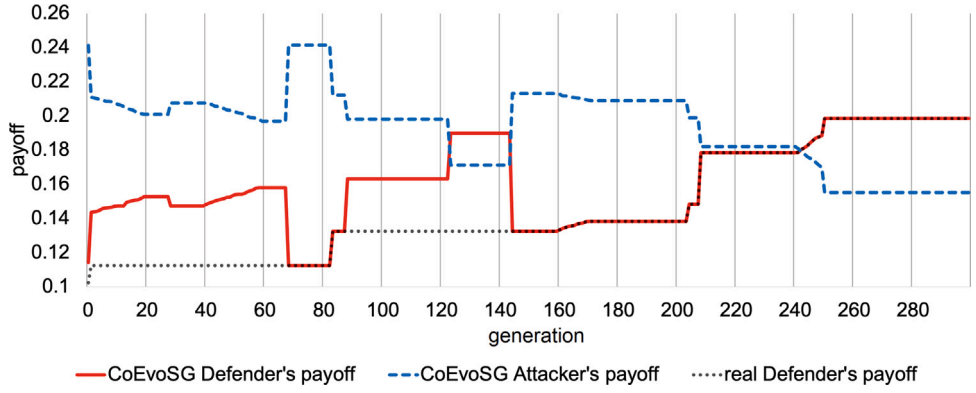


Fig. 9. Payoffs assigned to the best strategies from Defender's and Attacker's populations in consecutive generations in a typical CoEvoSG run. Vertical lines denote the moments of changing the population being currently developed and evaluated. Dotted line presents the real Defender's payoff, i.e. the one calculated against the optimal Attacker's response.

8.3. Attacker's population evaluation

It is generally desired that the Attacker's population is composed of optimal responses to all possible Defender's strategies. Assigning the average Attacker's payoff against all strategies from the Defender's population (or a subset of them) as a fitness value may be a weak approach because a given Attacker's strategy is usually strong only against a specific subset of the Defender's strategies. While such an Attacker's strategy should be preserved, the payoff averaging across all Defender's strategies will decrease a fitness of such a strategy, posing the 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 also exist certain weaker strategies. For those strategies, most of the Attacker's strategies will lead to a high Attacker's payoff and such an approach would not allow for distinguishing strong Attacker's strategies from the weak ones (the vast majority of the strategies will receive high fitness evaluation, as a maximum payoff against one of the weak Defender's strategies). This observation discredits the calculation of the maximum payoff with respect to all Defender's strategies. Fig. 11 confirms this reasoning. It shows the highest Defender's and Attacker's payoffs in consecutive generations with $N_{top} = N_D$. It can be noticed that in the Attacker's population rarely could be found a strong strategy that decreases the Defender's payoff, even though the real Defender's payoff (calculated against the respective optimal Attacker's response) is much lower. It means that the Defender's population cannot be fairly assessed this way.

On the other extreme, if the Attacker's fitness value was computed only against the best strategy from the Defender's population, it would lead to an oscillation 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. In such an approach the Attacker's population would lose diversity because all individuals would be optimized with respect to only one particular Defender's strategy. After the next period of evolution of the Defender's population the vast majority of the Attacker's strategies would again adjust to the new "target" Defender's strategy, and so on. Fig. 12 exemplifies this scenario and presents the maximum Defender's and Attacker's payoffs in the CoEvoSG runtime with $N_{top} = 1$. In such a case, clear oscillations can be observed. During the turn

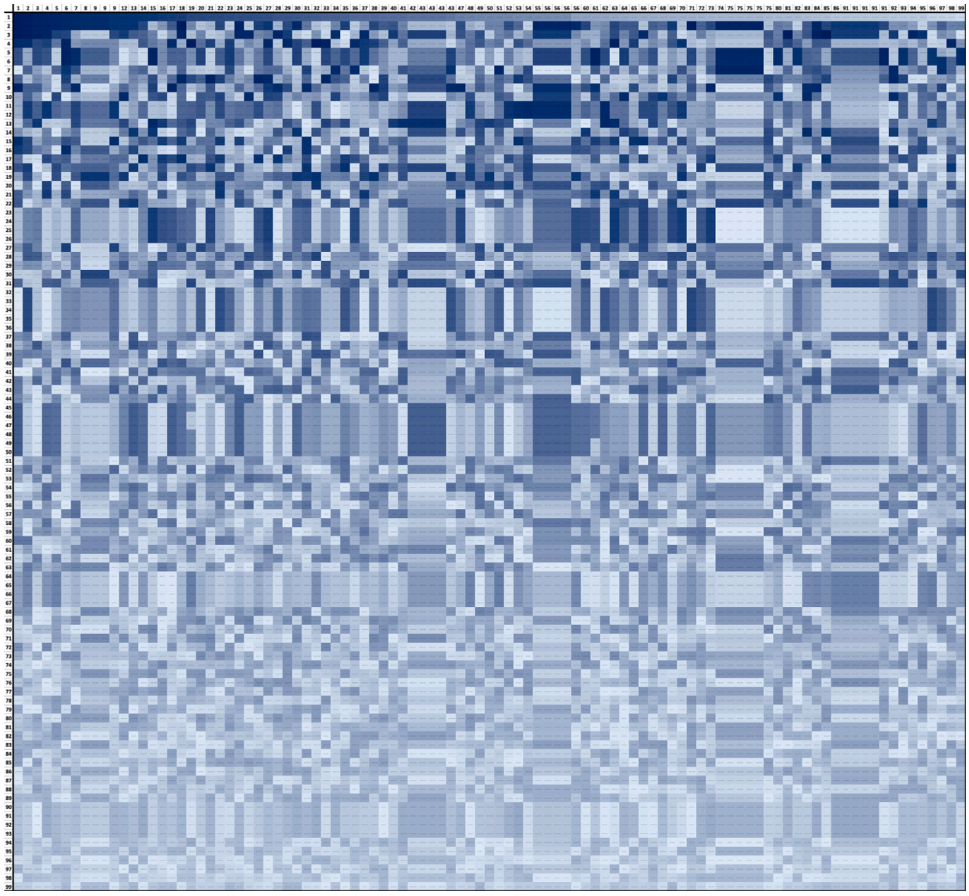


Fig. 10. Example interaction matrix between individuals from the Attacker's and the Defender's population after the last CoEvoSG generation. Each row represents one individual from the Defender's population, sorted by the expected payoff of the corresponding strategy. Each column represents one individual from the Attacker's population sorted by the Defender's payoff against the best individual from the Attacker's population. Each element of the matrix is colored according to the Defender's payoff obtained when playing strategies from a given row and column — the darker the color the higher the payoff.

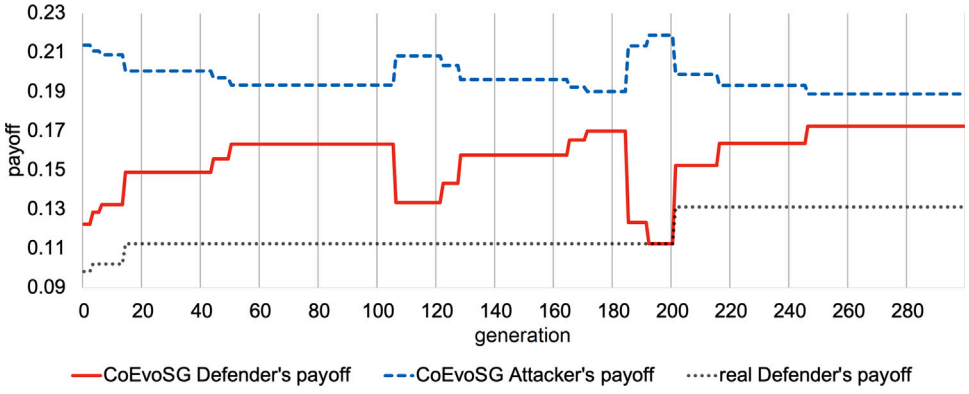


Fig. 11. Payoffs of the best strategies from the Defender's and the Attacker's populations in consecutive generations in a typical CoEvoSG run with $N_{top} = N_D$. Vertical lines denote the moments of changing the population being currently developed and evaluated. Dotted line presents the real Defender's payoff, i.e. the one calculated against the optimal Attacker's response.

of the Defender's population evolution the estimated Defender's payoff (calculated against the Attacker's population) increases quickly but the real Defender's payoff (calculated against the optimal Attacker's response) remains on the same level which means that the Attacker's population does not contain representative strategies (in particular the optimal one). However, during its evolution period the Attacker's population is able to quickly find a good response that decreases the Defender's payoff. At the same time, since it is optimized against the best Defender's strategy only, it loses diversity and the Defender's population in the next turn can easily find a strong alternative strategy (against such undifferentiated Attacker's responses).

Consequently, as stated in Section 6, an intermediate option was implemented, i.e. the Attacker's strategy fitness is the maximum of the Attacker's payoffs against the N_{top} highest fitted individuals from

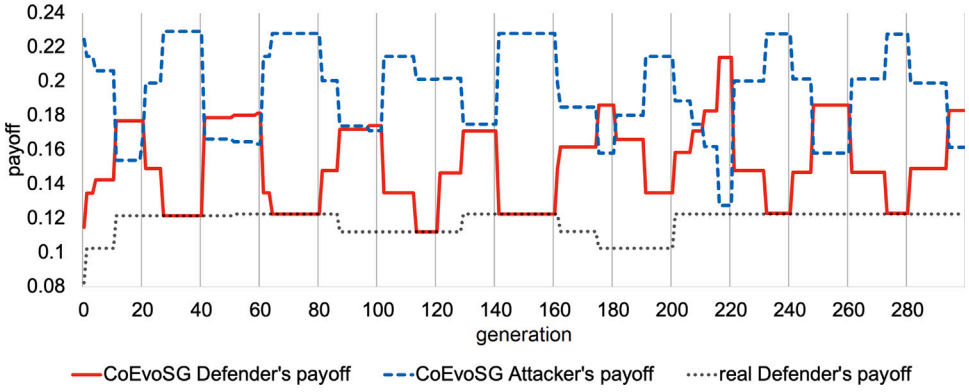


Fig. 12. Payoffs for the best strategies from the Defender's and the Attacker's populations in consecutive generations in a typical CoEvoSG run with $N_{top} = 1$. Vertical lines denote the moments of changing the population being currently developed and evaluated. Dotted line presents the real Defender's payoff, i.e. the one calculated against the optimal Attacker's response.

the Defender's population. Moreover, it was observed during the experiments that the addition of N_{random} random individuals from the Defender's population to the Attacker's evaluation procedure is beneficial and leads to even better results. A discussion on the parameters selection presented in Section 6 justifies particular N_{top} and N_{random} choices.

9. Conclusions

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

Experimental evaluation performed on three well-established game types with more than 800 test instances has proven the efficacy of the proposed method — in the majority of the test cases optimal solutions were found. The results are 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 the 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 is more, CoEvoSG is an *anytime* algorithm, i.e. is capable of returning a valid solution at any time of the execution process.

10. Future work

Our future work will concentrate on extending CoEvoSG to a more general adversarial framework including multiple heterogeneous Defenders and/or Attackers [31]. This can potentially be achieved by the maintenance of the corresponding multiple populations with complex interactions between them. In such a formulation not only populations of the opponents (Defenders and Attackers) may influence each other, but also populations on the same side (either Defenders or Attackers) can interact with each other in a competitive or collaborative manner.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Y. Guo, H. Zhang, L. Zhang, L. Fang, F. Li, A game theoretic approach to cooperative intrusion detection, J. Comput. Sci. 30 (2019) 118–126.
[2] A. Branitskiy, I. Kotenko, Hybridization of computational intelligence methods for attack detection in computer networks, J. Comput. Sci. 23 (2017) 145–156.
[3] A. Sinha, F. Fang, B. An, C. Kiekintveld, M. Tambe, Stackelberg security games: Looking beyond a decade of success, in: Proceedings of the 27th IJCAI Conference, 2018, pp. 5494–5501.
[4] A. Sinha, T.H. Nguyen, D. Kar, M. Brown, M. Tambe, A.X. Jiang, From physical security to cybersecurity, J. Cybersecurity 1 (1) (2015) 19–35.
[5] Y. Zhang, P. Malacaria, Bayesian Stackelberg games for cyber-security decision support, Decis. Support Syst. (2021) 113599.
[6] M. Jain, J. Tsai, J. Pita, C. Kiekintveld, S. Rathi, M. Tambe, F. Ordónez, Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service, Interfaces 40 (4) (2010) 267–290.
[7] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, G. Meyer, Protect: A deployed game theoretic system to protect the ports of the United States, in: Proceedings of the 11th AAMAS conference, 2012, pp. 13–20.
[8] F. Fang, T.H. Nguyen, R. Pickles, W.Y. Lam, G.R. Clements, B. An, A. Singh, M. Tambe, A. Lemieux, Deploying PAWS: Field optimization of the protection assistant for wildlife security, in: Proceedings of the 28th innovative applications of artificial intelligence conference, 2016, pp. 3966–3973.
[9] V. Conitzer, T. Sandholm, Computing the optimal strategy to commit to, in: Proceedings of the 7th ACM conference on electronic commerce, 2006, pp. 82–90.
[10] J. Karwowski, J. Mańdziuk, Stackelberg equilibrium approximation in general-sum extensive-form games with double-oracle sampling method, in: Proceedings of the 18th AAMAS conference, 2019, pp. 2045–2047.
[11] J. Karwowski, J. Mańdziuk, Double-oracle sampling method for Stackelberg equilibrium approximation in general-sum extensive-form games, in: Proceedings of the 34th AAAI Conference, vol. 34, 2020, pp. 2054–2061.
[12] J. Karwowski, J. Mańdziuk, A. Żychowski, F. Grajek, B. An, A memetic approach for sequential security games on a plane with moving targets, in: Proceedings of the 33rd AAAI Conference, vol. 33, 2019, pp. 970–977.
[13] A. Żychowski, J. Mańdziuk, Evolution of strategies in sequential security games, in: Proceedings of the 20th AAMAS Conference, 2021, pp. 1434–1442.
[14] A. Żychowski, J. Mańdziuk, Learning attacker's bounded rationality model in security games, in: Proceedings of the 28th ICONIP, vol. CCIS 1516, 2021, pp. 530–539.
[15] A. Żychowski, J. Mańdziuk, Coevolutionary approach to sequential Stackelberg security games, in: International Conference on Computational Science, Springer, 2022, pp. 103–117.
[16] A. Żychowski, J. Mańdziuk, E. Bondi, A. Venugopal, M. Tambe, B. Ravindran, Evolutionary approach to security games with signaling, in: Proceedings of the 31st IJCAI Conference, 2022, pp. 620–627.
[17] M. Breton, A. Alj, A. Haurie, Sequential Stackelberg equilibria in two-person games, J. Optim. Theory Appl. 59 (1) (1988) 71–97.

[18] B. Von Stengel, S. Zamir, Leadership With Commitment to Mixed Strategies, Tech. Rep. CDAM Research Report, 2004.
[19] P. Paruchuri, J.P. Pearce, J. Marecki, M. Tambe, F. Ordonez, S. Kraus, Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games, in: Proceedings of the 7th AAMAS Conference, 2008, pp. 895–902.
[20] B. Bošanský, J. Čermak, Sequence-form algorithm for computing Stackelberg equilibria in extensive-form games, in: Proceedings of the 29th AAAI Conference, 2015, pp. 805–811.
[21] J. Čermák, B. Bošanský, K. Durkota, V. Lisý, C. Kiekintveld, Using correlated strategies for computing Stackelberg equilibria in extensive-form games, in: Proceedings of the 30th AAAI Conference, 2016, pp. 439–445.
[22] L. Kocsis, C. Szepesvári, Bandit based Monte-Carlo planning, in: European Conference on Machine Learning, Springer, 2006, pp. 282–293.
[23] M. Świechowski, K. Godlewski, B. Sawicki, J. Mańdziuk, Monte Carlo Tree Search: A review of recent modifications and applications, Artif. Intell. Rev. (2023).
[24] A. Żychowski, J. Mańdziuk, A generic metaheuristic approach to sequential security games, in: Proceedings of the 19th AAMAS, 2020, pp. 2089–2091.
[25] L. Oakley, A. Oprea, QFlip: An adaptive reinforcement learning strategy for the flipit security game, in: International Conference on Decision and Game Theory for Security, Springer, 2019, pp. 364–384.
[26] M. Van Dijk, A. Juels, A. Oprea, R.L. Rivest, FlipIt: The game of "stealthy takeover", J. Cryptol. 26 (4) (2013) 655–713.
[27] D.J. Watts, S.H. Strogatz, Collective dynamics of 'small-world' networks, Nature 393 (6684) (1998) 440–442.
[28] J. Černỳ, B. Bošanskỳ, C. Kiekintveld, Incremental strategy generation for Stackelberg equilibria in extensive-form games, in: Proceedings of the 19th ACM conference on economics and computation, 2018, pp. 151–168.
[29] J. Karwowski, J. Mańdziuk, A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs, European J. Oper. Res. 277 (2019) 255–268.
[30] J. Mańdziuk, J. Karwowski, A. Żychowski, Simulation-based methods in multi-step Stackelberg security games in the context of homeland security, 2022, https://sg.mini.pw.edu.pl.
[31] J. Lou, A.M. Smith, Y. Vorobeychik, Multidefender security games, IEEE Intell. Syst. 32 (2017) 50–60.

Adam Żychowski, Ph.D., received his M.Sc. and Ph.D. degrees in Computer Science from the Warsaw University of Technology (WUT), Poland in 2015 and 2022, respectively. He is currently a research assistant at the Faculty of Mathematics and Information Science, WUT. His research interests include Game Theory and Computational Intelligence methods especially evolutionary algorithms and artificial neural networks.

Jacek Mańdziuk, Ph.D., D.Sc., IEEE Senior Member, received M.Sc. (Honors) and Ph.D. from the Warsaw University of Technology (WUT), Poland in 1989 and 1993, resp., and D.Sc. degree in Computer Science from the Polish Academy of Sciences in 2000. In 2011 he was awarded the title of Professor Titular. He is full professor at the Faculty of Mathematics and Information Science, WUT, Head of Division of Artificial Intelligence and Computational Methods, and Head of Doctoral Program in Computer Science at this faculty.

Prof. Mańdziuk was a recipient of the Fulbright Senior Research Award (UC Berkeley and ICSI Berkeley, USA) and the Robert Schuman Foundation Fellowship (CNRS, Besançon, France). Recently, he was a visiting professor at Nanyang Technological University (Singapore), University of New South Wales (Australia), Yonsei University (Korea) and University of Alberta (Canada).

His research interests include application of Computational Intelligence and Artificial Intelligence methods to games, dynamic and bilevel optimization problems, and human-machine cooperation in problem solving. He is also interested in the development of general-purpose human-like learning and problem-solving methods. He is the author of 3 books and 180+ research papers. For more information please visit http://www.mini.pw.edu.pl/~mandziuk