Evolution of Strategies in Sequential Security Games

Adam Żychowski
Faculty of Mathematics and Information Science
Warsaw University of Technology
a.zychowski@mini.pw.edu.pl

Jacek Mańdziuk
Faculty of Mathematics and Information Science
Warsaw University of Technology
j.mandziuk@mini.pw.edu.pl

ABSTRACT

In this paper we introduce a generic approach to solving Sequential Security Games (SGs) based on Evolutionary Algorithms. The method (named *EASG*) is general and largely game-independent, which allows for its application to a wide range of SGs with only slight adjustments. The efficacy of EASG is verified on 3 types of games from different domains: patrolling, surveillance, and cybersecurity. Comprehensive experiments performed on more than 300 test games demonstrate robustness and stability of EASG, manifested by repetitively achieving optimal or near-optimal solutions. The main advantage of EASG compared to alternative approaches to solving SGs is its time efficiency. EASG finds high-quality solutions with approximately linear time scalability and can therefore be applied to solving SG instances which are beyond capabilities of the state-of-the-art exact methods. Due to *anytime* characteristics, EASG is particularly well suited for time-critical applications.

KEYWORDS

Evolutionary algorithm; Security Games; Stackelberg equilibrium

ACM Reference Format:
Adam Żychowski and Jacek Mańdziuk. 2021. Evolution of Strategies in Sequential Security Games. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3-7, 2021*, IFAAMAS, 14 pages.

1 INTRODUCTION

Game theory (GT) offers mathematically justified solutions to many practical problems arising in various domains, e.g. economics [10], biology [31], politics [30] or social networks [38]. One of the fast-growing GT areas is security management (e.g. border controlling, cybersecurity, police surveillance, anti-terrorism policy, patrolling schedules, etc.). The majority of these applications base on the concept of Security Games.

Security Games (SGs) are typically two-player Stackelberg Games (StGs) [42]. One of the players - the *leader* (a.k.a. the *Defender* in SGs) commits to a strategy first and then the other player - the *follower* (a.k.a. the *Attacker* in SGs) chooses their strategy - based on the already announced leader's strategy. Such a sequence of decisions introduces asymmetry of information access, favoring the follower. The above concept fits certain real-life security-related scenarios in which the Attacker can surveil the guards (playing the Defender's role) patrolling some critical area and discover / deduct their patrolling strategy.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3-7, 2021, Online.

The list of successfully deployed real-world applications of SGs includes the scheduling system for Los Angeles International Airport canine patrols [15], the PROTECT system which randomizes schedules of US Coast Guard's resources in Boston harbour [36], the TRUSTS system for scheduling patrols for fare inspection in Los Angeles Metro system [45], or the PAWS system to prevent poaching and protecting wildlife in Queen Elizabeth National Park in Uganda [44]. Please consult a recent survey paper [37] for other examples.

In SGs the goal is to find a pair of players' strategies that correspond to the *Stackelberg Equilibrium* (StE) [27]. Since the Attacker is aware of the opponent's strategy in advance, their strategy is a consequence of the Defender's strategy (they choose a strategy which provides them the highest possible payoff). Hence, the problem of finding StE in SGs can be reduced to finding an optimal mixed Defender's strategy, which is still NP-hard [9].

On a general note, there are two major types of SGs solution methods: exact and approximate. Majority of the proposed exact approaches employ Mixed-Integer Linear Programming (MILP) [4, 15, 22, 41] and, consequently, suffer from poor time scalability which hinders their application beyond a certain level of game's complexity. Approximate methods, on the other hand, offer much better scalability but at the cost of finding close-to-optimal solutions [6, 17–19, 23, 26, 43]. This paper follows the latter approach and adopts Evolutionary Algorithms metaheuristic [2], which has proven efficient in solving other types of bi-level optimization problems [3, 8, 11].

Evolutionary Algorithms (EAs) are inspired by the process of (biological) evolution. Each member of a population in EA represents a candidate (valid) solution. In the iterative process of creating new generations three evolution-related operations: mutation, crossover and selection are performed. Despite the lack of rigorous proofs of convergence to the optimal solution, EAs - thanks to their experimentally proven efficiency - have been widely applied to a wide range of real-life optimization problems, e.g. [7, 13, 29, 46].

In the context of GT, Sefrioui et al. [35] applied co-evolutionary algorithm for computing the Nash equilibrium on a fluid dynamics problem and compared the results with Pareto Genetic Algorithms [12]. A similar approach, relying on colonial competitive algorithm [1], was proposed in [33]. There were also certain attempts of using EAs for finding StE [5, 34, 39], however, all of them were designed for simple one-step games. To our knowledge, except for the initial short version of this paper [47], the only application of EAs to sequential SGs was proposed in [20]. This method, however, is tailored to specific genre of SGs, i.e. SGs with targets moving on a plane [20].

1.1 Contribution

The main contribution of this paper is an introduction of the first evolutionary approach to finding near-optimal solutions of a wide

range of SGs in a computationally efficient manner. The method, abbreviated as EASG (Evolutionary Algorithm for Security Games), is domain-independent and flexible, i.e. applicable to various SGs formulations (one-step, multi-step, general-sum, with limited observability, with bounded rationality, etc.) with little adjustments only. Furthermore, due to *anytime* characteristics, EASG is specifically well suited for time-critical applications in SG domain.

EASG is tested on three types of SGs, where it demonstrates optimal or close-to-optimal performance, while outperforming the competitive approaches in terms of time scalability.

2 PROBLEM DEFINITION

We consider a sequential n-step StG with two players: the Defender (D) and the Attacker (A).

In StG a *pure strategy* of the player is defined as an assignment of one action to each *potentially reachable* state of the game. Please observe that the same action of the player may lead to several reachable states depending on the opponent's action selection [25]. Let's denote a set of all pure strategies of player p by Σ^p. A *mixed strategy* $\pi^p \in \Pi^p$, where Π^p is a set of all possible mixed strategies of player p, is a probability distribution over Σ^p.

In StG the Defender commits to their strategy $\pi^D \in \Pi^D$ (probability distribution of their pure strategies) and then the Attacker, being aware of π^D, determines their strategy π^A. Let's denote by $U^p(\pi^D, \pi^A)$ an expected utility value of player p as a result of the game played according to mixed strategies π^D and π^A. Stackelberg Equilibrium can be formally defined as a pair (π^D, π^A) satisfying the following equations:

$$BR(\pi^D) = \arg\max_{\pi^A \in \Pi^A} U^A(\pi^D, \pi^A) \tag{1}$$

$$\pi^D = \arg\max_{\bar{\pi}^D \in \Pi^D} U^A(\bar{\pi}^D, BR(\bar{\pi}^D)) \tag{2}$$

Equation (1) defines the Attacker's best (optimal) response $BR(\pi^D)$ to the Defender's strategy π^D while eq. (2) selects the best Defender's strategy against the optimal Attacker's response. In order to avoid ambiguity, if there exist more than one best Attacker's response, Strong Stackelberg Equilibrium (SStE) was defined [27] in which the Attacker, among all their best responses (with the same highest utility for them), selects the one with the highest Defender's utility, i.e. breaks ties in favor of the Defender. In this paper SStE version of StE is considered.

At the beginning of the game, both players choose their strategies (first the Defender and then the Attacker), which are played throughout the game, i.e. cannot be altered by the player is subsequent steps. In effect, in each time step $(\tau_1, \ldots, \tau_n)$ player p performs an action a_i^p ($p \in \{D, A\}, i = 1, \ldots, n$) according to their strategy, from the set of available actions $M(s_i^p)$, where s_i^p is a state of player p in time step τ_i. State s_i^p is determined by all previous player's actions, their initial position and the opponent's actions. Players perform actions (make moves) simultaneously, and are not aware of the opponent' current and past actions.

For each game state s there are four predefined payoffs $U^k(s)$ ($k \in \{A+, A-, D+, D-\}$) representing the Attacker's reward ($U^{A+}(s)$), their penalty ($U^{A-}(s)$), the Defender's reward ($U^{D+}(s)$) and their penalty ($U^{D-}(s)$), resp. Some of the states (usually those with high $U^{A+}(s)$ values) are distinguished and called the targets.

If in any time step $\tau_i (i = 1, \ldots, n)$:

- The Attacker and the Defender move to the same state (say s_i), then the game ends (the Attacker is intercepted) and the players receive payoffs $U^{A-}(s_i)$ and $U^{D+}(s_i)$, resp.
- The Attacker reaches any of the targets (say s_j) and is not intercepted, then the game ends and the respective payoffs are equal to $U^{A+}(s_j)$ and $U^{D-}(s_j)$.

Otherwise, the game lasts for n steps and ends with neutral payoffs.

3 STATE-OF-THE-ART APPROACHES

The vast majority of hitherto approaches to solving StG were focused on one-step game formulations [37] and either couldn't be applied to the multi-step case considered in this paper, or their application would be highly inefficient. In the literature there is just a handful of methods devoted to multi-step extensive-form StG, all of them developed in the last few years.

The first notable approach (BC2015) was introduced by Bošanský and Čermák in 2015 [4]. The authors extended a very popular MILP-based method DOBBS [32] to extensive-form games and introduced a novel algorithm for computing SStE, able to exploit the underlying structure of sequence-form games. The method is designed for non-zero-sum games and reduces the size of a linear program by transforming an extensive-form game into its equivalent sequence-form representation. In effect, the size of a linear program is reduced from exponential (as in DOBBS) to linear with respect to the game tree size, but still exponential with respect to the game length. An exact formulation of MILP utilized in BC2015 can be found in [4].

Another approach to the exact computation of SStE for two-player extensive-form general-sum games was proposed by Čermák et al. [41]. The method (henceforth referred to as C2016) utilizes a correlated version of SStE known as Stackelberg Extensive-Form Correlated Equilibrium (SEFCE). In SEFCE, the Defender can send signals to the Attacker who must follow these signals in their choice of the best response. C2016 relies on a linear program for computing SEFCE and then modifies it by iteratively restricting the signals the Defender can send to the Attacker and thus converging to SStE. In the experimental evaluation presented in [41], C2016 was superior to BC2015 in terms of time efficiency.

One of the recent heuristic approaches to SStE approximation was the Mixed-UCT method [16, 18] which incorporates a variant of Monte Carlo Tree Search, known as Upper Confidence Bounds applied to trees [24]. The authors [18] subsequently proposed another UCT-based approach - O2UCT [19, 21] which relies on a guided sampling of the Attacker's strategy space interleaved with finding (using double-oracle method [14]) a feasible Defender's strategy for which the just-sampled Attacker's strategy is the optimal response. Experimental evaluation of O2UCT shows the method's ability to find optimal or close-to-optimal solutions for various types of test games and better time-scalability than both the above mentioned exact MILP methods.

Another approximate method (denoted by CBK2018) [6] is a heuristic time-optimized MILP approach which constructs a smaller game tree representation with a specific abstraction structure called *gadgets*. The method significantly reduces the Defender's strategy space and therefore, brings down computation time requirements,

however at the cost of loosing theoretical MILP property of convergence to the optimal solution. A diminished game is solved with the C2016 method.

Four algorithms summarized above (two exact: BC2015, C2016 and two approximate: O2UCT, CBK2018) are state-of-the-art approaches to solving multi-step extensive-form StG and will be used as reference methods in the experimental evaluation of EASG.

4 EVOLUTIONARY ALGORITHM FOR SECURITY GAMES

Initially, a population of p_{size} individuals (chromosomes) is randomly generated. Each individual represents a potential solution - Defender's mixed strategy. In each subsequent generation, crossover and mutation operators are applied to randomly selected chromosomes from the current population, which is then followed by a selection procedure that promotes individuals to the next generation based on their evaluation (fitness). The fitness of a given chromosome is calculated as the Defender's payoff (when the Defender plays a mixed strategy represented by that chromosome) against an optimal Attacker's response (an Attacker's pure strategy yielding the highest payoff for them).

The above procedure is executed until the best (found so far) Defender's payoff does not change within n_c iterations or the limit for the number of generations n_g is exceeded. Please refer to supplementary material[1] for the algorithm's flowchart.

4.1 Chromosome representation

Each chromosome (individual) represents some Defender's mixed strategy (a candidate SStE solution) in the form of a vector of pure strategies π_i^q and their respective probabilities p_i^q:

$$CH_q = \{(\pi_1^q, p_1^q), \ldots, (\pi_{l_q}^q, p_{l_q}^q)\}, \quad \sum_{i=1}^{l_q} p_{l_q}^q = 1, \tag{3}$$

where q is the index of the chromosome in the population, l_q is the length of chromosome CH_q, i.e. the number of pure strategies included in the mixed strategy represented by that chromosome. A particular form of pure strategy depends on game specificity. In the most common case, a pure strategy is represented as a list of Defender's actions in consecutive time steps.

4.2 Initial population

Initial population contains solely pure strategies, i.e. $\forall_q\ l_q = 1\ \wedge\ p_1^q = 1$. These strategies are generated randomly in the following way. For each strategy, in each time step the next action is selected uniformly from all actions available in a given state. This procedure is independently executed for each chromosome in the initial population, i.e. p_{size} times.

4.3 Crossover

In the first step of crossover operation, a subset of $p_c \cdot p_{size}$ individuals are randomly selected from the population, where p_c is crossover rate. Then, individuals from this subset are randomly paired (in the case of an odd number of individuals, a randomly chosen one is removed). From each pair of individuals, one new *offspring* chromosome is created in the following way. All pure strategies from the *parent* chromosomes are merged into one mixed strategy with their probabilities halved. Repeating this operation unconditionally (in consecutive generations) would lead to chromosomes with large numbers of pure strategies with tiny probabilities. Thus, each pure strategy π_i^q in this newly created chromosome, except for the one with the highest probability, is removed with probability $(1 - p_i^q)^2$ (i.e. the lower the probability of a strategy the higher its chance for being deleted). Afterwards, probabilities of the remaining pure strategies are normalized so as to sum up to 1. The role of the crossover operator is to enhance the exploitation aspect of the EA by means of mixing strategies found so far.

4.4 Mutation

Mutation operator is applied to each chromosome independently with probability p_m. First, one pure strategy in the chromosome is randomly chosen. Then consecutively, starting from a randomly selected time step t_i up to the last time step t_n, an action in a considered time step $t_j, i \leq j \leq n$ is changed to an action uniformly chosen among all actions available in the corresponding game state. The role of the mutation operator is to boost exploration of the new areas of the search space.

4.5 Evaluation

The fitness of a chromosome is calculated as a Defender's payoff when they play a mixed strategy encoded in that chromosome. Since it is proven [9] that in StG there always exists at least one pure strategy of the follower which is their best response to the leader's strategy (pure or mixed), then it is sufficient to iteratively check all pure Attacker's strategies and select the one with the highest Attacker's payoff (additionally breaking ties in favor of the Defender - SStE condition). The Defender's payoff against the best Attacker's response found in the above described way is then computed and returned as a chromosome fitness value.

4.6 Selection

In the first step of the selection procedure, e individuals from the current population with the highest fitness values (including those created by mutation or crossover operations) are unconditionally promoted to the next generation population. These individuals are called *elite* and transmit the top ranked solutions found so far through consecutive generations.

Next, a *binary tournament* is iteratively executed until the next generation population is filled with p_{size} individuals. In each tournament, two chromosomes are sampled *with replacement* from the pool of currently available individuals (including those affected by crossover and/or mutation). The chromosome, within the pair, with higher fitness is promoted to the next generation with probability p_s. Otherwise, the lower-fitted one is promoted.

5 EXPERIMENTAL SETUP

5.1 Benchmark games

Properties of EASG were tested on 3 sets of multi-step games with variable characteristic: Warehouse Games [18], Search Games [4], and FlipIt Games [40], which were previously used in the literature

[1] www.mini.pw.edu.pl/~mandziuk/PRACE/AAMAS-21.pdf

for evaluation of various state-of-the-art SG approaches. While all 3 types of games are defined on graphs, their rules, properties and related challenges are vastly different, which makes them a truly diverse benchmark set.

5.1.1 Warehouse Games. The Warehouse Games (WHG) proposed in [18] are played on graphs which mimic the layouts of warehouse buildings. Vertices represent storage rooms and corridors. Attacker and Defender start in fixed (different) vertices. In each time step, each player can move to any of the neighboring vertices (directly connected through an edge) or stay in the currently occupied vertex. The game ends in any of the following three cases:

- **a)** players meet in the same vertex, which means interception of the Attacker - a negative payoff for the Attacker and a positive one for the Defender,
- **b)** the Attacker reaches one of the distinguished vertices (targets) and is not intercepted - the Attacker receives a positive payoff and payoff of the Defender is negative,
- **c)** none of the two above cases takes place in any of the n time steps - the payoff equals 0 for both playing sides.

The payoff structure is non-zero-sum. A more detailed description of these benchmark games is presented in [18]. The number of steps in WHG varies between 3 and 8 that leads to the game trees of the sizes from 10^2 to 10^8 nodes. For each number of steps $t = 3, \ldots, 8$, 25 games were tested (150 different games in total). All test games were downloaded from the project website [28].

In WHG an action is in the form of a decision about the next node to be visited, hence in EASG application to WHG a chromosome contains a list of nodes to be visited in consecutive time steps.

5.1.2 Search Games. The Search Games (SEG) [4] 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 Defender controls several units and furthermore, these units cannot move freely on the entire graph but each of them has a subset of vertices assigned which it 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 they visit 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 the WHG - 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.

In total, 90 games with 4, 5, and 6 time steps, played on 3 different graph structures proposed in [4] were used in the evaluation process. Attacker's and Defender's penalties were set to -1 and for each graph structure 10 random distributions of rewards were sampled from interval $[1, 2]$.

In order to be able to solve SEG instances the EASG chromosome representation needs to be extended. Formally, let's denote by $\mathcal{L}^u_{vt}$ a list of nodes visited in consecutive time steps by unit u in the case of trace discovery in node v at time step t and by $\mathcal{L}^u_\emptyset$ a list of nodes visited by unit u in the case of no trace discovery during the whole gameplay. Then, the Defender's pure strategy π^D in SEG is of the following form: $\pi^D = \{\mathcal{L}^u_{vt}, u \in D_u, v \in V, t \in \{1, \cdots, n\}\} \cup \{\mathcal{L}^u_\emptyset, u \in D_u\}$, where D_u is a set of Defender's units, V is a set of graph vertices, n is the number of game steps.

Each pure strategy represents one of possible compound scenarios. For each Defender's unit, such a scenario is related to either discovering or not (by that unit) a trace in any particular vertex, in any particular time step during the entire game. A set of such pure strategies with assigned probabilities defines a mixed strategy for the Defender, represented by a chromosome (cf. eq. (3)).

5.1.3 FlipIt Games. The FlipIt Games (FIG) [40] were initially proposed for evaluation of CBK2018 method [6] and refer to cybersecurity settings in which the Attacker attempts to gain control over certain resources (e.g. servers, hubs, PCs) and the Defender may take actions to restore their control of the infected units.

The game is played on a directed graph for a fixed number of time steps. In each step, each of the players selects a node he/she attempts to take control of (to *flip the node*). Only some of the nodes (so-called entry nodes) are publicly accessible and the Attacker is obliged to start gaining control and penetrate the network from one of these nodes (in other words the only possible Attacker's action in the first step is selection of one of the entry nodes). A flip action is successful if the player controls also at least one of the preceding nodes (or it is the entry node with no predecessors) and the current owner of the selected node does not take the flip action in this node in the same time step. Successful flip action results in gaining control on the flipped node.

At the beginning of the game, all nodes are controlled by the Defender. Each node has assigned a reward for controlling it, and a cost of its flip attempt. The final player's payoff is the sum of rewards from all nodes controlled by them after each time step (nodes controlled in multiple steps count multiple times), decreased by the costs of all flip attempts (either successful or failed).

In the experiments, 60 FIG instances played on 3 different graph structures proposed in [6] were used. For each graph, 5 different payoffs structures were randomly drawn. The number of time steps was set to 3, 4, 5 or 6. The experiments were performed in *No-Info* game variant [6] in which players have no information about the results of their actions - they are not aware of whether or not their flip action succeed and, therefore, their strategy is independent of the opponent's actions.

FIG are solved with the same EASG settings as in the case of WHG, with no specific adaptation. In particular, strategies are represented in a chromosome as lists of nodes which the Defender would attempt to flip in consecutive time steps.

5.2 EASG parametrization

EASG parameters were tuned on 50 WHG with 5 and 6 steps (excluded from the final tests). The initially selected sets of parameter values are presented in Table 1. EASG was run 5 000 times, in each case with a random selection of parameters (from Table 1) and a randomly chosen game. Figures. 1a- 1d present Defender's payoff and computation time for the respective tested parameter averaged across all trails. Based on these outcomes the final parameter values (bolded in Table 1) were chosen, taking into account both the expected payoff and the running time.

Table 1: Parameter values selected for the tuning process. Finally recommended values are bolded.

parameter	symbol	value
population size	p_{size}	10, 20, 50, **100**, 200, 500, 1000, 2000, 5000
# generations	n_g	**1000**
# generations with no improvement	n_c	**20**
mutation rate	p_m	0, 0.1, 0.2, 0.3, 0.4, **0.5**, 0.6, 0.7, 0.8, 0.9, 1
crossover rate	p_c	0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, **0.8**, 0.9, 1
selection pressure	p_s	0.6, 0.7, 0.8, **0.9**, 0.95, 1
# elite	e	**2**

Figure 1a shows that an increase of *population size* results in higher Defender's payoffs since more individuals can explore strategy space and consequently the higher number of potential solutions is considered. For small values ($10 \leq p_{size} \leq 100$) a steep increase of the expected payoff can be observed which subsequently flattens. Computation time scales approximately linearly with p_{size}.

A relation between the Defender's payoff and *mutation rate* is presented in Fig. 1b. Mutation is a key element of the EA process and without it ($p_m = 0$) the Defender's payoff deteriorates drastically. On the other hand, too high mutation rate also entails payoff decrease, since the process is too disordered - the majority of strategies are randomly changed in each generation. Computation time naturally grows along with the mutation rate increase due to the increasing number of chromosome modifications, but the increase is not as significant as in the case of population size (Fig. 1a).

Similarly to mutation, *crossover* is recognized as a critical operator in EA and its removal ($p_c = 0$) causes significant payoff decrease (Fig. 1c). At the same time, differences among payoffs for all other (positive) crossover rate values are not significant. Clearly, the higher the p_c the more crossover operations, and consequently the higher computation time, but the time increase is not steep.

Selection pressure $p_s < 0.5$ would mean that weaker individuals are more likely to be chosen, hence only $p_s > 0.5$ were tested. The best results were obtained for $p_s = 0.9$ (see Fig. 1d). Since p_s does not affect the number of EASG operations, computation time remains approximately constant across all tested values of p_s.

6 EXPERIMENTAL EVALUATION OF EASG

EASG is evaluated from four perspectives: convergence, results quality, stability, and time scalability. All presented results were obtained in 30 independent runs per game instance, with parameters and implementation setup discussed in the previous section. In total EASG assessment was based on 9 000 trials (150 WHG, 90 SEG and 60 FIG, each tested 30 times). All tests were run on Intel Xeon Silver 4116 @ 2.10GHz with 256GB RAM. The source code can be downloaded from our project website [28].

6.1 Convergence

Figure 2 visualizes two typical convergence characteristics of EASG. The payoffs of random strategies in the initial population are usually centered around one (left figure) or two (right figure) values. In subsequent generations the payoffs are more scattered but, generally, the mean Defender's payoff increases in time, which means that the entire population moves towards the areas with higher payoffs. At the same time, some low-payoff individuals exist in practically all generations due to the use of mutation operator that leads to the exploration of new strategies.

In none of the experiments EASG attained the maximum number of generations n_g (set to 1 000) and in all runs terminated because of the other stopping condition - no improvement of the best-found solution in consecutive n_c generations (set to 20).

Figure 3a presents a histogram of the numbers of generations in all experiments. The maximum value of 181 was achieved in one of the 8 step WHG instances. In more than half of the cases, the number of generations was less than 30, which means that the final solution was found within the first 10 generations. This observation supports the claim about fast and stable convergence of EASG.

It can be proven that multiple application of proposed operators can, in principle, lead to any arbitrary solution. In other words, given enough time, any mixed strategy can potentially be achieved through repeated application of these operators, independently of the initial population selection.

6.2 Results quality

EASG as an approximate algorithm does not guarantee finding optimal solutions. However, experimental evaluation shows that in the majority of the cases the method yields optimal SStE strategies, and in the rest of them, a distance to the optimal solution is narrow.

In order to calculate the optimal solutions for considered game instances, both exact methods (BC2015 and C2016) were run with the limit of 200h per game. If none of the methods were capable of finding the SStE strategy for the Defender within the allotted time, the game was excluded from the evaluation of results quality. In effect, 100 WHG (out of 150), 60 SEG (out of 90) and 45 FIG (out of 60) instances were used in the EASG quality assessment.

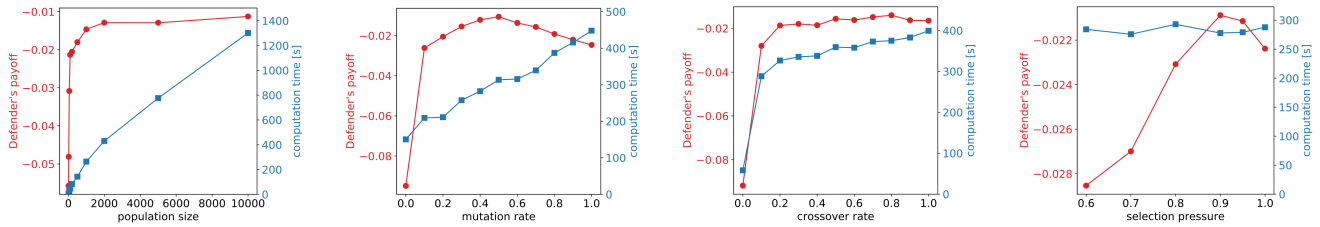
A histogram of the differences between the optimal solution and the one provided by EASG in all runs, across all tested game instances with known optimal solutions is presented in Fig. 3b.

Please note that FIG final payoffs are summed over all controlled vertices, in all time steps. Hence, to assure a direct comparability with the results achieved for other game types, they were normalized to $[-1; 1]$, i.e. the lowest and the highest possible Defender's payoffs (computed by the exact method) were mapped to -1 and 1, resp.

6.2.1 WHG. In the case of WHG, both exact methods were able to calculate the SStE for 100 games with $3 - 6$ time steps. In all tests with larger games ($7 - 8$ steps) the solution could not be reached due to extensive time requirements. In 72 out of 100 games EASG obtained optimal solutions. The mean difference between the EASG results and the optimal ones equaled 0.0013, and the highest difference equaled 0.0127 (3.7% of the possible payoff range).

6.2.2 SEG. For SEG, optimal solutions are known for 60 games with 4 and 5 steps out of which EASG found optimal strategies in 28 cases (47%). The average divergence from the optimal results equaled 0.0253 with the highest value of 0.0955 (12.2% of the possible payoff range).

In spite of satisfying results, it should be noted that they are noticeably worse than those for WHG. The reason for that is most



(a) Population size (p_{size}). (b) Mutation rate (p_m). (c) Crossover rate (p_c). (d) Selection pressure (p_s).

Figure 1: The average Defender's payoff (circles) and computation time (squares) with respect to the main steering parameters of EASG, calculated during the parameter tuning phase.

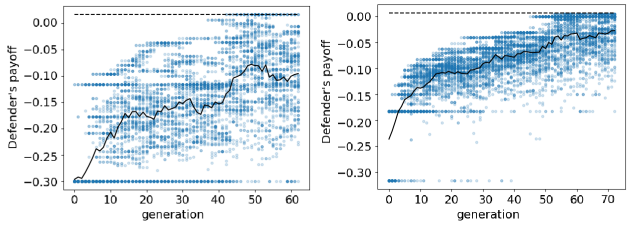
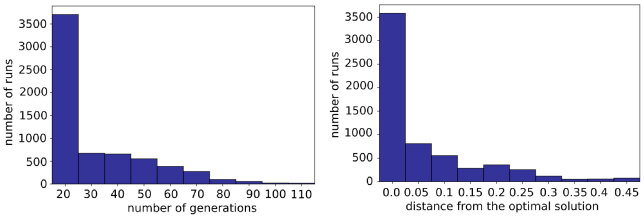


Figure 2: Typical EASG performance characteristics. Figures present payoffs of all chromosomes in consecutive generations for two sample games, resp. Dashed lines denote optimal solutions (Defender's payoffs in SStE), solid curves - the average payoffs across all individuals (in a given generation).



(a) (b)

Figure 3: (a): The number of generations (across all tests) before EASG termination. (b): Differences between optimal Defender's payoffs and payoffs obtained by EASG (for all games with known optimal solutions).

probably much larger strategy representation - SEG require strategies for several (more than one) Defender's units, additionally with specific variants for the cases of discovering traces, so the effective strategy search space is much wider than that of WHG.

In order to verify this hypothesis, additional experiments with bigger populations were executed. If the reason for a relative deterioration of results is indeed SEG's larger strategy space and reacher solution representation, increasing population size should cause a more extensive search of the strategy space and yield better results. The outcomes of these experiments, presented in Figure 4a, confirm that with greater populations EASG was able to find optimal solutions for a higher percentage of SEG instances, albeit at the cost of computation time increase.

6.2.3 FIG. The third set of games - FIG are also recognized as a challenging task for SGs solution methods due to their extensive search space. Please note that FIG game trees grow very fast even for small graphs since in each step (except for the first one in which some restrictions on the Attacker's choice apply) any of the two players can attempt to flip any of graph vertices (not only a successor of his/her current position). For a graph with n_v vertices, in each non-initial step, each of the players has n_v available actions, so the number of nodes in a game tree exceeds n_v^{2n-1}, where n is the number of steps.

Despite large strategy space, EASG managed to achieve optimal solutions in 73% of the cases (exact methods were capable of finding solutions for 45 test games, out of which EASG yielded the same solutions for 33 games). The average divergence from the optimal results (normalized in a way described above) equaled 0.0087 with the highest value of 0.0321 (6.4% of the possible payoff range).

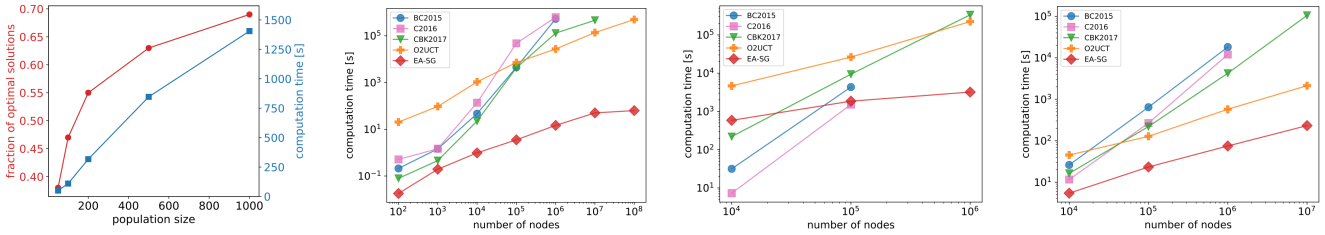
The quality of results for FIG is comparable to that for WHG and higher than in the case of SEG. Again, the reason for this results disparity is attributed to a much complex chromosome representation in SEG. While in FIG and WHG solutions (Defender's mixed strategies) are encoded straightforwardly, in SEG each chromosome contains separate strategies for each unit and each possible trace discovery scenario (depending on both the vertex and time step). Hence, finding an optimal strategy in such a complex space representation is more difficult in a stochastic strategy changing process performed by EASG operators.

6.3 Comparison with O2UCT and CBK2017.

Table 2 compares EASG with the two other state-of-the-art methods. CBK2017 and EASG reach optimal solutions in comparable number of runs, however, in terms of the average and the worst outcomes, EASG visibly outperforms CBK2017. O2UCT is the most repeatable method and achieves the lowest average distance, albeit has the smallest number of optimal solution runs.

6.4 Stability

EASG, typically for EA methods, is highly non-deterministic (the initial population, both operators, as well as the selection procedure contain random factors). Hence, the sole ability to obtain optimal solutions, discussed in the previous subsection, is not sufficient for



(a) (b) Warehouse Games. (c) Search Games. (d) FlipIt Games.

Figure 4: (a) A fraction of SEG instances for which EASG found optimal solutions (circles), and computation time requirements (squares) with respect to the population size. (b)-(d) Comparison of EASG time scalability vs state-of-the-art methods. Please note the logarithmic scale on both axes.

Table 2: Comparison of 3 heuristic approaches in terms of the average and the highest distance to the optimal solution, and the fraction of games ended with an optimal result.

Method	*WHG*			*SEG*			*FIG*		
	Avg	Max	%	Avg	Max	%	Avg	Max	%
CBK2017	0.072	0.325	**77%**	0.084	0.276	43%	0.014	0.127	63%
O2UCT	**0.001**	**0.006**	74%	**0.015**	**0.058**	37%	**0.004**	**0.012**	58%
EASG	**0.001**	0.013	72%	0.025	0.096	**47%**	0.009	0.032	**73%**

the comprehensive EASG assessment. An equally important aspect of the algorithm is its ability to reproduce good results.

In order to check EASG stability, standard deviations of Defender's payoffs in 30 runs were computed for all games (including those omitted in the previous subsection due to the lack of exact solutions) across 30 runs for each game. For 45% of games, standard deviation was equal to 0 (perfect stability). The mean standard deviation equaled 0.0059 with the maximal value 0.1629 (36.7% of the possible payoff range).

Among 133 games for which the optimal solutions were found, the best solution was repeated in all 30 runs in 82 (62%) cases. For 96 of these games (72%) the optimal solution was found in more than 90% of runs. At the other extreme, in 23 cases (17%) the optimal strategy was achieved only once. The above figures indicate that, despite a certain level of randomness, the method offers relatively high stability and is capable of regularly reproducing the optimal solution in the majority of the tested games.

6.5 Time scalability

Parameter tuning presented in Section 5.2 indicates that time performance of EASG strongly depends on selected steering parameters. Hence, their adjustment allows establishing the expected balance between computation time and quality of results.

Figure 4 compares time efficiency of EASG (with parameters setting listed in Table 1) vs four state-of-the-art algorithms summarized in Section 3. First, the games of a given type (separately WHG, SEG and FIG) were divided into subsets of instances with pairwise comparable game tree sizes (after rounding pairwise equal to the nearest power of 10). Then, for each subset the running times of all game instances belonging to that subset were averaged and plotted. As we mentioned in Section 6.2, due to exceeding time limit of 200 hours per trial, for the biggest games the results of exact methods (BC2015 and C2016) could not be plotted.

Computation times of all five methods grow exponentially with respect to the game length. However, in the case of MILP-based algorithms (BC2015, C2016 and CBK2018) the exponents are higher than in the case of approximate approaches (O2UCT and EASG). Overall, EASG demonstrates the highest time efficiency among the tested methods.

6.6 Additional convergence analysis

In this section we discuss certain internal convergence properties of EASG. Due to space limits we present only general observations referring to the average behavior of the method. More detailed data is discussed in supplementary material.

First of all, it should be stressed that convergence characteristics strongly depends on a game instance. In some simple games EASG finds optimal strategy very fast, already in the 2nd generations, whereas in some difficult instances it takes more than 100 generations to find a stable point. The average number of generations was equal to 34.

A fitness value for the best individual mostly changes in the first 5 iterations. In the majority of the cases, around the 5th iteration mixed strategy in the best chromosome already contains all pure strategies which will make up for the final result. In almost all runs the average population fitness value increased in time, i.e. a population as a whole was improving. Standard deviation of fitness in consecutive generations slowly increased (along with increasing population diversity) and then stabilized.

The number of pure strategies per chromosome usually increases linearly in the first 10-20 generations and then stabilizes. The average number of pure strategies in the final solutions is equal to 5.2. The maximum observed number of them was 37 and the minimum was 1 (a single pure strategy with probability 1).

The mean mutation success rate, i.e. a fraction of mutations which led to fitness improvement was strongly correlated with a game type and equaled on average 32% for WHG, 2% for SEG and 14% for FIG instances. The main reason for this diversity can be attributed to the fact that the number of all (potential) pure strategies and their related payoffs significantly differ between game types. Since mutation introduces a random change into a mixed strategy its real impact (success rate) strongly depends on particular game rules. We discuss this issue further in supplementary material.

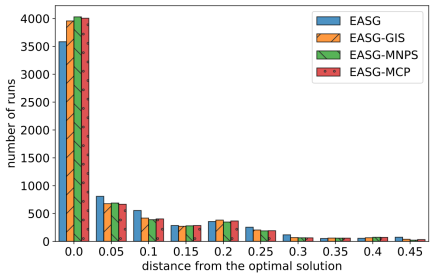


Figure 5: Histogram of differences between optimal Defender's payoffs and payoffs obtained by baseline EASG and its optimized versions.

7 EASG MODIFICATIONS

EASG formulation is intentionally generic so as to make the method widely applicable to various types of SGs. Consequently, the EA implementation in EASG is rather canonical. In this section we discuss several enhancements to EASG which are *tailored to SG domain, though still generic, i.e. independent of the particular game formulation.*

(A) *Greedy initial strategies (*EASG-GIS*).* In EASG pure strategies in the initial population are selected randomly. However, a non-random starting point may potentially lead to better outcomes. To this end, the baseline population generation method is augmented in the following way. First, $10p_{size}$ pure strategies are generated randomly and evaluated. Next, p_{size} of them with the highest fitness (Defender's payoff) constitute an initial population.

(B) *Mutation with new pure strategies (*EASG-MNPS*).* Pure strategies in EASG population can change only as a result of mutation. However, EASG lacks a mechanism for adding new pure strategies. Thus, we proposed a new type of mutation (MNPS) which is performed right after the baseline mutation described in Section 4.4. MNPS adds a randomly selected pure strategy to the mixed strategy encoded in the chromosome. The probability of this newly added pure strategy is sampled from the unit interval. Afterward, all probabilities in the modified mixed strategy are normalized. Initial experiments with MNPS showed that its single application rarely leads to chromosome improvement. Hence, in the tested setup it is tried multiple times until the resultant chromosome has higher fitness or the maximum number of mutation trials m_{limit} is reached.

(C) *Mutation with changing probability (*EASG-MCP*).* In EASG probabilities in mixed strategy are changed only as a result of crossover operation and their values are always halved. For the sake of higher diversity, we propose another mutation operator (MCP) which assigns a new uniformly chosen probability to a uniformly sampled pure strategy in a chromosome. Subsequently, all probabilities in that modified mixed strategy are normalized. Similarly to MNPS, this procedure is tried repeatedly until a better-fitted chromosome is obtained or m_{limit} is reached.

All 3 above-described modifications ((A)-(C)) were tested on the same set of games described in Section 6. m_{limit} was set to 50. A comparison of EASG with EASG-GIS, EASG-MNPS and EASG-MCP presented in Figure 5 shows that all three modified versions of EASG outperform the baseline algorithm by a clear margin. While EASG obtained optimal solutions in 72% of WHG, 47% of SEG and 73% of FIG instances, the respective outcomes for EASG-GIS: (74%, 54%, 74%), EASG-MNPS: (76%, 55%, 77%) and EASG-MCP: (75%, 51%, 74%) indicate that the main improvement is observed for the hardest class of games - SEG.

(D) *Other modifications.* One can also improve the results by adding a *memetic* local optimization phase to EASG, tailored to a particular game definition. For instance, in SEG one can compute the best Attacker's response strategy, detect these components (L^u_{vt}) which affect the resulting payoff and restrict mutation application only to them. Such an optimization increases the rate of optimal solutions from 47% (baseline mutation) to 64%. We did not include any domain-specific optimizations in EASG for the sake of keeping the method generic. Further discussion of potential EASG modifications is presented in the supplementary material.

8 CONCLUSIONS

The paper presents a novel Evolutionary Algorithm approach to solving sequential Security Games. The method explores the space of Defender's strategies by means of evolving a population of candidate strategies in order to find Strong Stackelberg Equilibrium. The method is general and easy to adapt to other types of SGs, and, more broadly, to other problems that consist in finding SSE.

Experimental evaluation, performed on games of 3 different types, with 300 instances in total, proved robustness and time efficiency of EASG. In more than half of the tested game instances the optimal solutions were found, and for the remaining ones the average distance to the optimal solution was very low.

The experiments demonstrate that EASG scales in time visibly better than the state-of-the-art approaches, and, consequently, can be applied to bigger games (in terms of the size of a game tree or Defender's strategy space). Despite the lack of rigorous time and quality convergence proofs, which is a common situation for EA-based approaches, promising experimental results make us believe that the method presents a viable alternative to state-of-the-art algorithms when solving larger and more complex sequential SGs.

Due to its iterative construction, EASG is well suited for time-critical applications. Its execution can be interrupted at any moment, and still a valid solution (the best one found so far) will be returned.

There is still room for improvement of EASG. Three generic modifications introduced in Section 7 visibly enhanced the baseline implementation, indicating the research potential along these lines.

ACKNOWLEDGMENTS

The project was funded by the National Science Centre, grant number 2017/25/B/ST6/02061.

REFERENCES

[1] Esmaeil Atashpaz-Gargari and Caro Lucas. 2007. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In *2007 IEEE Congress on Evolutionary Computation*. IEEE, 4661–4667.
[2] Thomas Bäck, David B Fogel, and Zbigniew Michalewicz. 1997. *Handbook of evolutionary computation*. CRC Press.
[3] Jonathan F Bard. 2013. *Practical bilevel optimization: algorithms and applications*. Vol. 30. Springer Science & Business Media.
[4] Branislav Bošanský and Jiří Čermak. 2015. Sequence-Form Algorithm for Computing Stackelberg Equilibria in Extensive-Form Games. In *Proceedings of the*

Twenty-Ninth AAAI Conference on Artificial Intelligence. 805–811.

[5] José-Fernando Camacho-Vallejo, Álvaro Eduardo Cordero-Franco, and Rosa G González-Ramírez. 2014. Solving the bilevel facility location problem under preferences by a Stackelberg-evolutionary algorithm. *Mathematical Problems in Engineering* 2014 (2014).

[6] Jakub Černỳ, Branislav Boỳanskỳ, and Christopher Kiekintveld. 2018. Incremental strategy generation for Stackelberg equilibria in extensive-form games. In *Proceedings of the 2018 ACM Conference on Economics and Computation*. ACM, 151–168.

[7] Carlos A Coello Coello and Gary B Lamont. 2004. *Applications of multi-objective evolutionary algorithms*. Vol. 1. World Scientific.

[8] Benoît Colson, Patrice Marcotte, and Gilles Savard. 2007. An overview of bilevel optimization. *Annals of operations research* 153, 1 (2007), 235–256.

[9] Vincent Conitzer and Tuomas Sandholm. 2006. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM conference on Electronic commerce*. ACM, 82–90.

[10] James W Friedman. 1986. *Game theory with applications to economics*. Vol. 87. Oxford University Press New York.

[11] Abhishek Gupta, Jacek Mańdziuk, and Yew-Soon Ong. 2015. Evolutionary multitasking in bi-level optimization. *Complex & Intelligent Systems* 1, 1-4 (2015), 83–95.

[12] Jeffrey Horn, Nicholas Nafpliotis, and David Goldberg. 1994. A niched Pareto genetic algorithm for multiobjective optimization. In *Proceedings of the first IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, Vol. 1. Citeseer, 82–87.

[13] Yong Hu, Kang Liu, Xiangzhou Zhang, Lijun Su, EWT Ngai, and Mei Liu. 2015. Application of evolutionary computation for rule discovery in stock algorithmic trading: A literature review. *Applied Soft Computing* 36 (2015), 534–551.

[14] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. 2011. A double oracle algorithm for zero-sum security games on graphs. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 327–334.

[15] Manish Jain, Jason Tsai, James Pita, Christopher Kiekintveld, Shyamsunder Rathi, Milind Tambe, and Fernando Ordónez. 2010. Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. *Interfaces* 40, 4 (2010), 267–290.

[16] Jan Karwowski and Jacek Mańdziuk. 2015. A new approach to security games. In *International Conference on Artificial Intelligence and Soft Computing*. Springer, 402–411.

[17] Jan Karwowski and Jacek Mańdziuk. 2016. Mixed Strategy Extraction from UCT Tree in Security Games. In *Proceedings of the Twenty-Second European Conference on Artificial Intelligence* (The Hague, The Netherlands) *(ECAI'16)*. IOS Press, NLD, 1746–1747.

[18] Jan Karwowski and Jacek Mańdziuk. 2019. A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research* 277, 1 (2019), 255–268.

[19] Jan Karwowski and Jacek Mańdziuk. 2019. Stackelberg Equilibrium Approximation in General-Sum Extensive-Form Games with Double-Oracle Sampling Method. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2045–2047.

[20] Jan Karwowski, Jacek Mańdziuk, Adam Żychowski, Filip Grajek, and Bo An. 2019. A memetic approach for sequential security games on a plane with moving targets. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. 970–977.

[21] Jan Karwowski and Jacek Mańdziuk. 2020. Double-oracle sampling method for Stackelberg Equilibrium approximation in general-sum extensive-form games. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. 2054–2061.

[22] Christopher Kiekintveld, Manish Jain, Jason Tsai, James Pita, Fernando Ordóñez, and Milind Tambe. 2009. Computing optimal randomized resource allocations for massive security games. In *Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 689–696.

[23] Christopher Kiekintveld and Vladik Kreinovich. 2012. Efficient approximation for security games with interval uncertainty. In *2012 AAAI Spring Symposium Series*. 42–46.

[24] Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *European conference on machine learning*. Springer, 282–293.

[25] Harold W Kuhn. 1950. Extensive games. *Proceedings of the National Academy of Sciences of the United States of America* 36, 10 (1950), 570.

[26] VS Anil Kumar, Rajmohan Rajaraman, Zhifeng Sun, and Ravi Sundaram. 2010. Existence theorems and approximation algorithms for generalized network security games. In *2010 IEEE 30th International Conference on Distributed Computing Systems*. IEEE, 348–357.

[27] George Leitmann. 1978. On generalized Stackelberg strategies. *Journal of optimization theory and applications* 26, 4 (1978), 637–643.

[28] Jacek Mańdziuk, Jan Karwowski, and Adam Żychowski. 2019. *Simulation-based methods in multi-step Stackelberg Security Games in the context of homeland security. https://sg.mini.pw.edu.pl.*

[29] Jacek Mańdziuk and Adam Żychowski. 2016. A memetic approach to vehicle routing problem with dynamic requests. *Applied Soft Computing* 48 (2016), 522–534.

[30] Nolan McCarty and Adam Meirowitz. 2007. *Political game theory: an introduction*. Cambridge University Press.

[31] Geoffrey A Parker and J Maynard Smith. 1990. Optimality theory in evolutionary biology. *Nature* 348, 6296 (1990), 27–33.

[32] Praveen Paruchuri, Jonathan P Pearce, Janusz Marecki, Milind Tambe, Fernando Ordonez, and Sarit Kraus. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 895–902.

[33] Ramin Rajabioun, Esmaeil Atashpaz-Gargari, and Caro Lucas. 2008. Colonial competitive algorithm as a tool for Nash equilibrium point achievement. In *International Conference on Computational Science and Its Applications*. Springer, 680–695.

[34] Masatoshi Sakawa, Ichiro Nishizaki. 2000. Computational methods through genetic algorithms for obtaining Stackelberg solutions to two-level mixed zero-one programming problems. *Cybernetics & Systems* 31, 2 (2000), 203–221.

[35] M Sefrioui and J Perlaux. 2000. Nash genetic algorithms: examples and applications. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*, Vol. 1. IEEE, 509–516.

[36] Eric Shieh, Bo An, Rong Yang, Milind Tambe, Craig Baldwin, Joseph DiRenzo, Ben Maule, and Garrett Meyer. 2012. Protect: A deployed game theoretic system to protect the ports of the united states. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1*. International Foundation for Autonomous Agents and Multiagent Systems, 13–20.

[37] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. 2018. Stackelberg Security Games: Looking Beyond a Decade of Success. In *IJCAI*. 5494–5501.

[38] Marco Slikker and Anne Van den Nouweland. 2012. *Social and economic networks in cooperative game theory*. Vol. 27. Springer Science & Business Media.

[39] Thomas Vallée and Tamer Başar. 1999. Off-line computation of Stackelberg solutions with the genetic algorithm. *Computational Economics* 13, 3 (1999), 201–209.

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

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

[42] Heinrich Von Stackelberg. 1934. *Marktform und gleichgewicht*. Springer.

[43] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. 2019. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*. 1401–1408.

[44] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. 2014. Adaptive resource allocation for wildlife protection against illegal poachers. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multiagent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 453–460.

[45] Zhengyu Yin, Albert Xin Jiang, Matthew P Johnson, Christopher Kiekintveld, Kevin Leyton-Brown, Tuomas Sandholm, Milind Tambe, and John P Sullivan. 2012. Trusts: Scheduling randomized patrols for fare inspection in transit systems. In *Proceedings of the Twenty-Fourth Conference on Innovative Applications of Artificial Intelligence*. 59.

[46] Adam Żychowski, Abhishek Gupta, Jacek Mańdziuk, and Yew Soon Ong. 2018. Addressing expensive multi-objective games with postponed preference articulation via memetic co-evolution. *Knowledge-Based Systems* 154 (2018), 17–31.

[47] Adam Żychowski and Jacek Mańdziuk. 2020. A Generic Metaheuristic Approach to Sequential Security Games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2089–2091.

— *SUPPLEMENTARY MATERIAL* — AAMAS'2021

In this appendix additional analysis of the EASG method and a discussion on its potential enhancements are provided. We start with presentation of a general flowchart of the method (Fig. 6) and examples of crossover and mutation operations (Figs. 7 and 8, respectively). Then, in Section B we discuss certain EASG convergence properties in more detail, and in Section C propose potential directions for EASG improvement, mainly by means of adjusting evolutionary operators.

A EASG FLOW-CHART AND EVOLUTIONARY OPERATORS

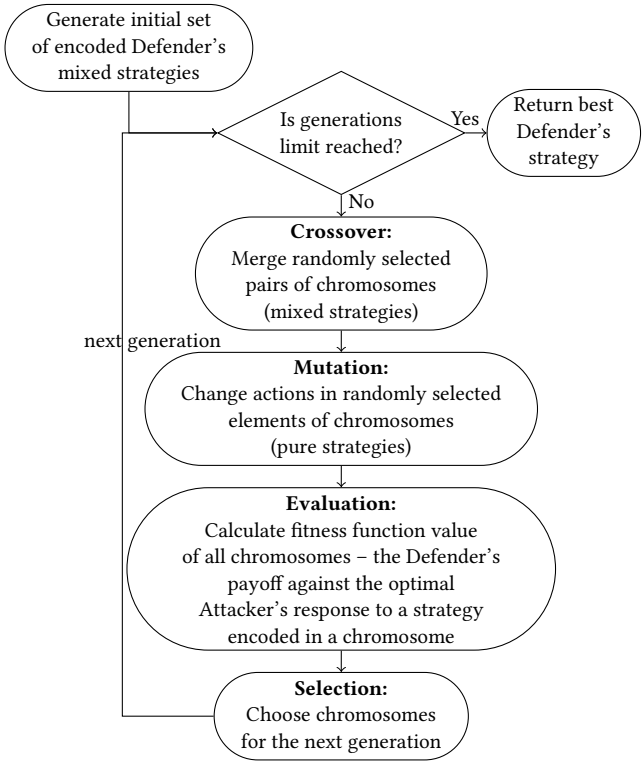


Figure 6: An overview of the EASG method.

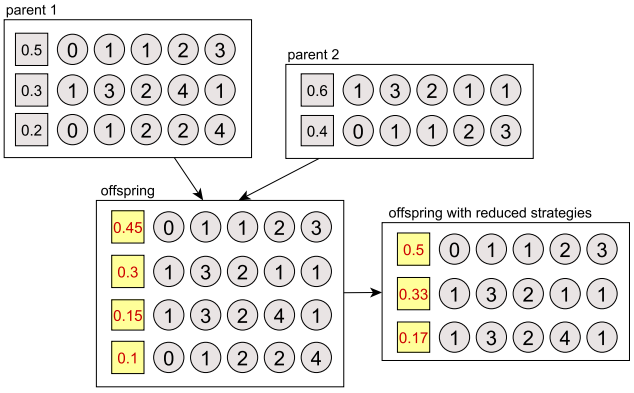


Figure 7: An example of crossover operation. Each row of circles represents a sequence of actions in consecutive time steps (Defender's pure strategy). Squares denote probabilities assigned to the respective strategies. First, an offspring chromosome inherits all pure strategies from both of its parents (with halved probabilities). Then, some strategies are potentially removed (in the example it is the one with the lowest probability) and probabilities of the remaining strategies are normalized (to sum up to 1).

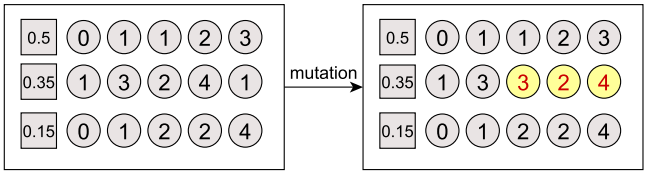


Figure 8: An example of mutation operation. Each row of circles represents a sequence of actions in consecutive time steps (Defender's pure strategy). Squares denote probabilities assigned to the respective strategies. In the presented case, the second pure strategy undergoes mutation, starting from the third time step.

B PROPERTIES OF THE BASELINE EASG

B.1 Convergence pattern

A typical pattern of EASG convergence is presented in Figure 9. The mean fitness value increases in consecutive generations which means that the entire population moves towards the areas with higher payoffs. At the same time, the lowest payoff is quite far away from the average population value calculated across all generations. This property stems from an application of mutation operator which introduces random changes into strategies and may potentially deteriorate (the assessment of) an individual. Both properties (the average fitness improvement and low minimal fitness) are generally considered desired properties of an evolutionary algorithm, contributing to the diversity and the strength of the population.

The improvement of the fittest individual (with maximal Defender's payoff) is most significant during the first few generations. In subsequent generations only small adjustments are observed.

B.2 Convergence speed

Figure 10 illustrates a distribution of the number of generations required before a stop condition is satisfied. In the vast majority of experiments this value stays below 30. Since the stop condition is triggered if the best result does not improve in 20 consecutive generations, in most cases the final solution was effectively found within the first 10 generations. The average number of generations equaled 34 with a maximum of 181.

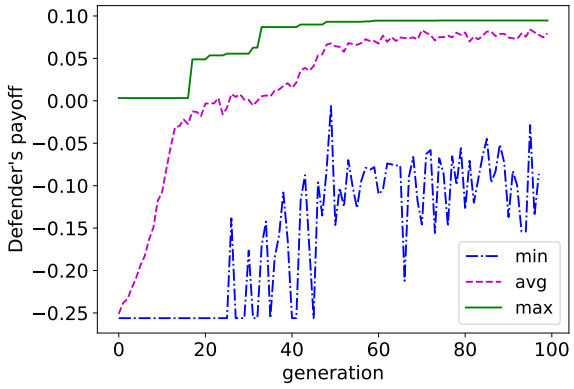


Figure 9: Maximal, minimal and average fitness values (Defender's payoffs) in a population in a typical run.

B.3 Stability

Stability of EASG solutions is illustrated in Figure 11 which, for each game type, shows a distribution of standard deviations of payoffs, each of them computed based on 30 independent runs for a given game instance. In the vast majority of cases, the standard deviation is 0 which proves high repeatability of the method, despite its stochastic nature.

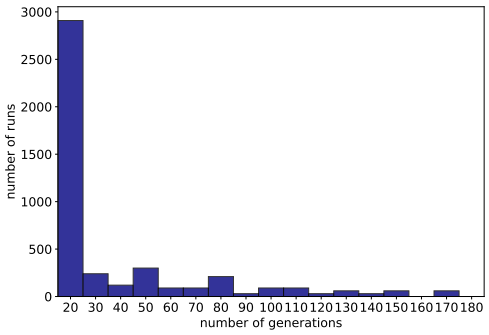
B.4 Chromosome sizes

Figure 12 demonstrates the average length of mixed strategies encoded by chromosomes, i.e. the average number of pure strategies constituting a mixed strategy in a chromosome, in 3 exemplar runs. In the initial phase a complexity of an average mixed strategy increases approximately linearly, reaching a plateau around the 50th generation. This stabilization is an effect of a particular form of crossover operator. In the case of the higher number of pure strategies in a chromosome their probabilities are relatively smaller which increases a chance for their deletion during crossover operation.

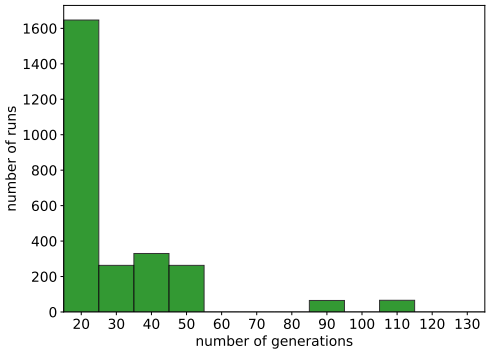
The number of pure strategies in the optimal (theoretical) solution strongly depends on a particular game instance and in the considered benchmark varies between 2 and 28. An experimental distribution of this value aggregated by game type is presented in Figure 13. The majority of the resulting mixed strategies are composed of 2-4 pure strategies. In rare situations their number exceeds 14.

B.5 Mutation success rates

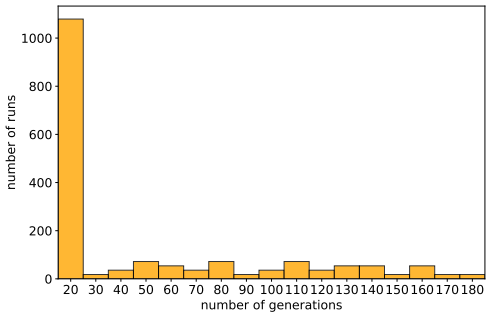
An interesting property, apparently strongly correlated with game type, is mutation success rate, i.e. a fraction of mutations that led to fitness improvement. Its distribution is presented in Figure 14. For Warehouse Games mutation improves a chromosome quite frequently - roughly 30 – 35% of mutations ends up with better fitted individuals. In the remaining two genres of games (FlipIt Games and Search Games) this rate is much lower. This variability can be explained by the differences in game rules and strategy space sizes.



(a) Warehouse Games.



(b) Search Games.



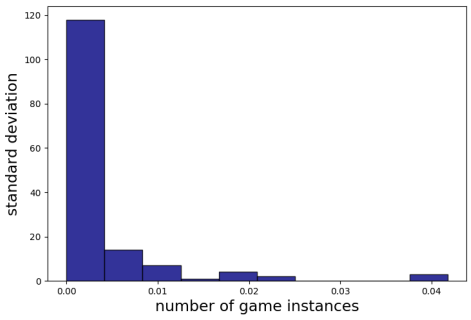
(c) FlipIt Games.

Figure 10: Histograms of the numbers of generations in all experiments.

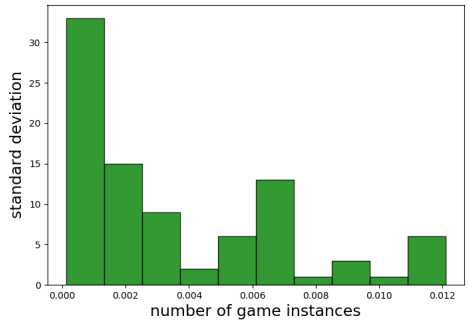
For instance, in SEG some pure Defender's strategies refer to the case of trace discovery (cf. Section 5.1.2 of the main paper), however, if the Attacker's strategy is to erase traces, then changes in these pure Defender's strategies would have no influence on the actual payoffs. In effect, a mutation applied to such strategies would not affect the fitness value of an individual.

C EASG ENHANCEMENTS

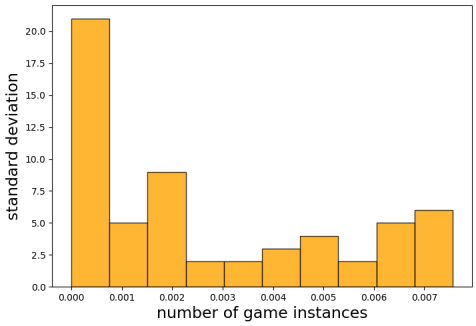
Several modifications to the baseline EASG formulation have been tested during the experimental assessment of the method. All of them preserve a generic character of the proposed algorithm. Each EASG variant was tested on the same set of benchmarks as the



(a) Warehouse Games.



(b) Search Games.



(c) FlipIt Games.

Figure 11: A distribution of standard deviations of payoffs in all experiments aggregated by game type.

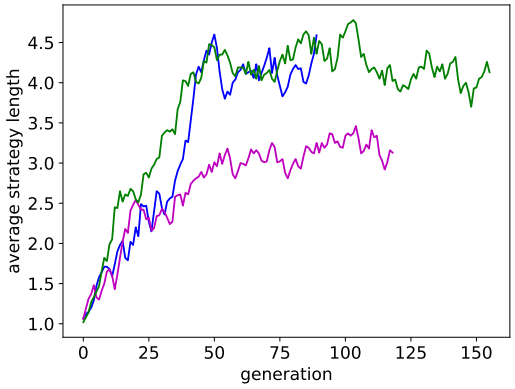
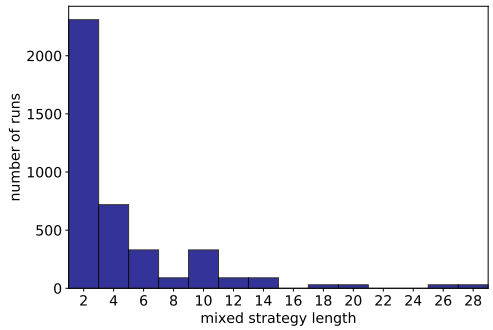


Figure 12: Average number of pure strategies in all chromosomes in a population, in 3 sample runs.

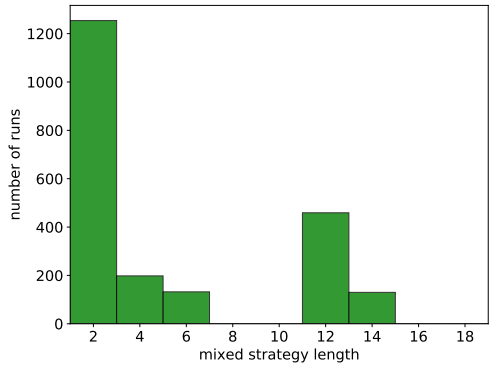
baseline EASG version. In this section we first briefly describe all considered modifications and then present and discuss the results of their application.

C.1 Tested variants of EASG

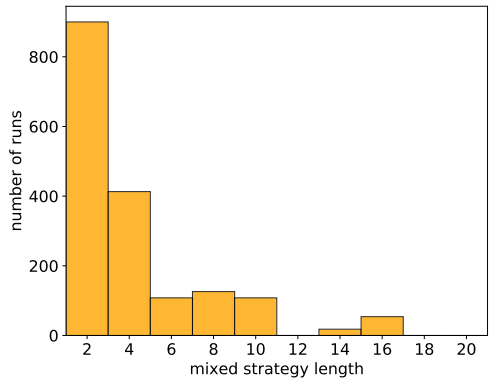
- **MNPS** - *mutation with new pure strategies* - mutation operator is extended with a new action: a randomly selected pure strategy is added to a chromosome with uniformly sampled probability. The chromosome is reverted to its previous form if its fitness deteriorates. The mutation is repeated until a better solution is found or predefined limit of trials m_{limit} is reached.
- **MCP** - *mutation with changing probability* - mutation operator is extended with a new action: a probability of randomly selected pure strategy is uniformly changed. The chromosome is reverted to its previous form if its fitness deteriorates. The mutation is repeated until a better solution is found or predefined limit of trials m_{limit} is reached.
- **MSP** - *mutation with switching probability* - mutation operator is extended with a new action: probabilities of two randomly chosen pure strategies are switched. The chromosome is reverted to its previous form if its fitness deteriorates. The mutation is repeated until a better solution is found or predefined limit of trials m_{limit} is reached.
- **MDPS** - *mutation with deleting pure strategy* - mutation operator is extended with a new action: a randomly chosen pure strategy is removed. The chromosome is reverted to its previous form if its fitness deteriorates. The mutation is repeated until a better solution is found or predefined limit of trials m_{limit} is reached.
- **MNPS**$_1$ - *MNPS modification without repetition* - a randomly chosen new pure strategy is added (regardless of the resulting impact on the individual fitness).
- **MCP**$_1$ - *MCP modification without repetition* - a probability of a randomly selected pure strategy is changed only once.
- **MSP**$_1$ - *MSP modification without repetition* - probabilities of two randomly selected pure strategies are switched (regardless of the resulting impact on the individual fitness).
- **MDPS**$_1$ - *MDPS modification without repetition* - a randomly selected pure strategy is removed (regardless of the resulting impact on the individual fitness).
- **MDPS**$_W$ - *deleting the weakest pure strategy* - mutation operator is extended with a new action: deleting pure strategy with the lowest payoff.
- **CWP** - *crossover with payoff consideration* - in crossover operator, instead of removing pure strategies based on their probabilities (as in the baseline EASG crossover) pure strategies are removed with probability inversely proportional to their payoff.



(a) Warehouse Games.

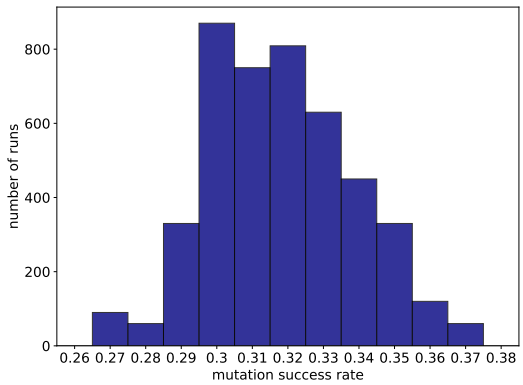


(b) Search Games.

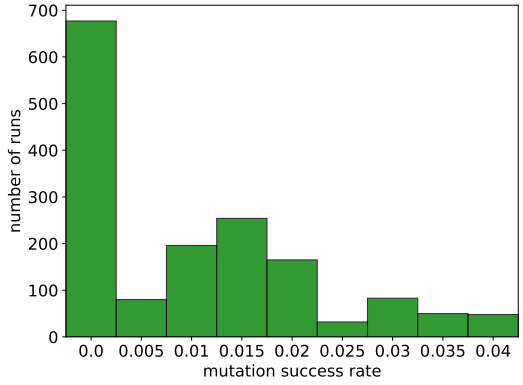


(c) FlipIt Games.

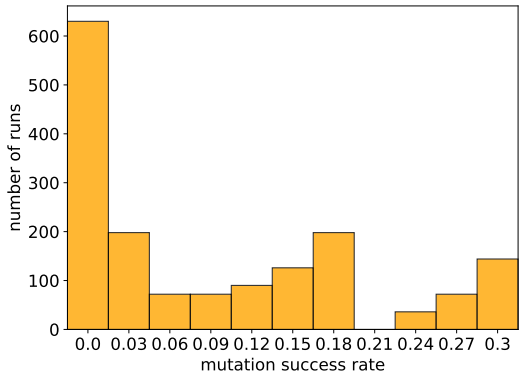
Figure 13: Histograms of the numbers of pure strategies in final solutions calculated for all experiments.



(a) Warehouse Games.



(b) Search Games.



(c) FlipIt Games.

Figure 14: Histograms of mutation success rates (fractions of mutations that led to solution improvement) calculated for all experiments, averaged across all generations in a given run.

- **GIS** - *greedy initial strategies* - initial pure strategies are selected from a wide set of pure strategies; p_{size} pure strategies with the greatest payoffs constitute the initial population.
- **MWPS** - *mutation of weakest pure strategy* - mutation is applied only to a pure strategy with the lowest payoff.
- **MWPS**$_P$ - *mutation of weakest pure strategy proportional* - mutation is applied to randomly chosen pure strategy, but the probability of its selection is inversely proportional to the expected payoff obtained after its application.

- **NIG** - *new individuals in a generation* - a bunch of new pure strategies are added to the population in each generation.

	EASG	MNPS	MCP	MDPS	MSP	CWP	GIS	$MNPS_1$	MCP_1	$MDPS_1$	MSP_1	MWPS	$MWPS_P$	NIG	$MDPS_W$
WHG	0.015	**0.016**	**0.016**	0.013	0.016	0.014	0.015	0.013	0.013	0.008	0.014	0.015	0.013	0.015	0.005
SEG	0.048	**0.139**	0.131	0.033	0.108	0.059	0.074	0.099	0.052	0.018	0.050	0.099	0.046	0.094	0.019
FIG	0.031	0.036	**0.037**	0.026	0.037	0.031	0.034	0.030	0.032	0.018	0.034	0.029	0.031	0.031	0.016

Table 3: The average Defender's payoffs in various EASG variants. The first column (EASG) contains reference results obtained for the baseline algorithm. The best results for each game type are bolded.

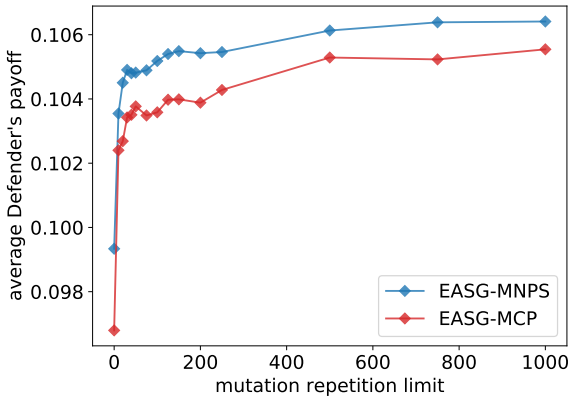


Figure 15: Influence of mutation repetition limit m_{limit} on the average Defender's payoff for MNPS and MCP modifications.

C.2 Performance

Table 3 presents the average Defender's payoffs in all above-introduced EASG variants, separately for each game type, with m_{limit} set to 50. Based on the results the examined modifications can be divided into 3 groups:

- better than baseline EASG - i.e. MNPS, MCP, MSP, and GIS. In the case of WHG and FIG games the advantage is moderate but for SEG instances the gain is significant;
- worse than baseline EASG - i.e. MDPS, $MDPS_1$, $MDPS_w$ (methods which remove pure strategies);
- similar to baseline EASG - the rest of the methods obtained results comparable to those of EASG.

In all cases, variants with repeated mutations produced better outcomes than their counterparts without repetition (denoted by subscript 1). Repeating mutation until a better individual is generated makes mutation a greedy process and increases the mutation success rate approximately 2-3 times.

Figure 15 confirms the usefulness of multiple-trial mutations. The greater the value m_{limit} the higher the expected Defender's payoffs obtained. The gain is particularly significant in low range of the limit i.e. $m_{limit} < 50$. Starting from $m_{limit} \approx 400$ the payoff curves flatten.