Optimized mutation operator in evolutionary approach to Stackelberg Security Games

Adam Żychowski[0000–0003–0026–5183], **Jacek Mańdziuk**[0000–0003–0947–028X]

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

Abstract. *In this paper, we introduce several mutation modifications in Evolutionary Algorithm for finding Strong Stackelberg Equilibrium in sequential Security Games. The mutation operator used in the state-of-the-art evolutionary method is extended with several greedy optimization techniques. Proposed mutation operators are comprehensively tested on three types of games with different characteristics (in total over* 300 *test games). The experimental results show that application of some of the proposed mutations yields Defender's strategies with higher payoffs. A trade-off between the results quality and the computation time is also discussed.*
Keywords: *Security Games, Stackelberg Equilibrium, Evolutionary Computation*

1. Introduction

Evolutionary Algorithms (EAs) are popular and powerful population-based metaheuristic optimization methods. Their effectiveness strongly depends on designed evolutionary operators: mutation, crossover, and selection [1]. One of the potential improvements can be achieved by adding some local optimization techniques – either as a separate algorithm step or as a part of the evolutionary operators [2].

In this study, the above claim is verified with respect to the evolutionary algorithm for sequential Stackelberg Security Games (SSGs) [3], by means of an introduction and experimental evaluation of various greedy optimizations for the mutation operator. Some of the proposed modifications lead to better results and superior Defender's strategies.

2. Problem definition

Sequential SSGs are played by two players: the Defender (D) and the Attacker (A). Each game is composed of m time steps and each player chooses an action to be performed (simultaneously) in each time step. A player's *pure strategy* σ_P ($P \in \{D, A\}$) is a sequence of their actions in consecutive time steps: $\sigma_P = (a_1, a_2, \ldots, a_m)$. The set of all possible pure strategies of player P is denoted by Σ_P. A probability distribution $\pi_P \in \Pi_p$ over Σ_P is the player's *mixed strategy*, where Π_p is the set of all mixed strategies for player P.

For any pair of strategies (π_D, π_A) the expected payoffs for the players are denoted by $U_D(\pi_D, \pi_A)$ and $U_A(\pi_D, \pi_A)$. The goal of the game is to find the *Strong Stackelberg Equilibrium* (SSE), i.e. a pair of strategies (π_D, π_A) satisfying the following conditions:

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

The first equation chooses the best Defender's strategy π_D under the assumption that the Attacker always selects the best response strategy ($BR(\pi_D)$) to the Defender's committed strategy. If there exists more than one optimal Attacker's response (with the same highest Attacker's payoff), the Attacker selects the one with the highest corresponding Defender's payoff, i.e. breaks ties in favor of the Defender [4].

Both players choose their strategies at the beginning of a game (first the Defender and then the Attacker) and the strategies cannot be altered during the course of the game. The problem of finding SSE has been proven to be NP-hard [5].

3. Evolutionary Algorithm

The Evolutionary Algorithm for Stackelberg Games (EASG) [6] aims to optimize the Defender's payoff by evolving a population of Defender's mixed strategies. Initially, EASG creates a population of pure Defender's strategies selected at random. The population evolves over successive generations until the stopping criterion is met. Four operations are applied in each generation: crossover, mutation, evaluation, and selection.

Crossover randomly selects two individuals from the population and combines their pure strategies by halving their probabilities and merging them into a single chromosome. The resultant chromosome is simplified by deleting some of its pure strategies, with the probability of deletion being inversely proportional to their probabilities.

The mutation operator randomly selects a pure strategy encoded in the chromosome and modifies it, starting from a randomly selected time step. New actions are drawn from the set of all feasible actions in a given game state.

Each individual in the population is then assigned a fitness value, which represents the expected Defender's payoff. This requires finding the optimal Attacker's response to the mixed Defender's strategy encoded in the chromosome. EASG accomplishes this by iterating over all possible Attacker's pure strategies and selecting the one with the highest Attacker's payoff [6].

In the selection phase, individuals with higher Defender's payoffs have a higher likelihood of being chosen for the next generation. This is achieved through a binary tournament in which two chromosomes are repeatedly selected with return, and the one with the higher fitness value is promoted with a certain probability (the *selection pressure*). Additionally, a set of chromosomes with the highest fitness function value is unconditionally copied to the next generation to preserve the best solutions found so far (the *elite* mechanism).

EASG is a generic framework which can be applied to various SSGs. For instance, it has been successfully applied to games with moving targets [7], signaling games [8], or games that assume bounded rationality of the Attacker [9, 10].

4. Mutation modifications

In EASG the mutation changes random actions from randomly selected pure strategy. We observed that this approach rarely leads to individuals with higher fitness function. For some types of games it is less than 6% of mutation executions, while the recommended literature standard is 20% - *one-fifth rule* [11]. **In order to improve the mutation impact, we propose and test various mutation implementations, other than the baseline [6], which may potentially improve Defender's strategies encoded in chromosomes.**

We test 11 different types of mutation modifications, which are briefly described below. Rather than entirely replacing the original mutation operator in EASG we propose that the new type of mutation be applied (replace the original one) with a probability equal to 0.5. All other EASG operators and parameters remain the same as reported in [6]. Depending on the number of mutations applied, we generally distinguish two cases:

(1) The mutation is applied only once and its result is preserved regardless of the resulting impact on the individual fitness (this approach was used in EASG). All such variants will be denoted with a subscript 1, e.g. MNPS_1,
(2) The mutation is repeated until a better solution is found or a predefined limit of trials n is reached. After each trial, the chromosome is reverted to its previous form if its fitness deteriorates. Such variants will be denoted by subscript n, e.g. MNPS_n. In the experiments, $n = 50$ was used.

In either case, if the encoded mixed strategy probabilities have changed as a result of applied mutation, they are normalized to sum up to 1. The following mutation enhancements have been tested:
- $\mathbf{EASG}_n$ - EASG algorithm with repeated mutation.
- $\mathbf{MANPS}_1$, $\mathbf{MANPS}_n$ - *mutation adds new pure strategy* - a uniformly selected pure strategy is added to a chromosome, with a uniformly sampled probability.
- $\mathbf{MCP}_1$, $\mathbf{MCP}_n$ - *mutation changes probability* - a probability of randomly selected pure strategy is uniformly changed.
- $\mathbf{MSP}_1$, $\mathbf{MSP}_n$ - *mutation switches probability* - probabilities of two randomly chosen pure strategies are switched.
- $\mathbf{MDPS}_1$, $\mathbf{MDPS}_n$ - *mutation deletes pure strategy* -a randomly chosen pure strategy is removed.
- **MCWPS** - *mutation changes the weakest pure strategy* - mutation is applied only to a pure strategy with the lowest payoff.
- **MDWPS** - *mutation deletes the weakest pure strategy* - pure strategy with the lowest payoff is deleted.

5. Results and Conclusions

All mutation variants have been tested on 3 types of Security Games: Warehouse Games (WHG) [12], Search Games (SEG) [13], and FlipIt Games (FIG) [14]. The same game instances were also used for EASG evaluation [6]. Please refer to [6] for a detailed description of the rules and characteristics of the games.

Table 1 shows experimental results averaged over 30 independent runs and all game instances – 150 WHG, 90 SEG, and 60 FIG. The biggest improvement is observed for SEG. This may be attributed to the fact that SEG is the most complex type of game, with the largest search space. Variants with mutation repetitions yield higher results improvements, but also lead to a significant (approximately tenfold) increase in computation time. The reason for that is frequent evaluation

Table 1. The average and standard deviation values of the Defender's payoff and the computation time for various mutation operators. The best results are **bolded**. Results that are better than the baseline version of the algorithm (EASG) are underlined. In cases where the difference between the baseline version (EASG) and a given variation is statistically significant (according to the Wilcoxon test with p-value < 0.05), the result is highlighted with a gray background.

	Defender's payoff			Computation time [s]		
	WHG	SEG	FIG	WHG	SEG	FIG
EASG	**0.017** ±0.001	0.108 ±0.006	0.031 ±0.002	152±6	2534±150	328±20
EASG$_n$	**0.017** ±0.001	0.135 ±0.008	**0.037** ±0.003	1206±84	21913±1264	3051±224
MANPS$_1$	0.014 ±0.001	0.059 ±0.004	0.031 ±0.002	156±8	2548±119	313±11
MANPS$_n$	0.016 ±0.001	**0.139** ±0.013	0.036 ±0.002	1366±62	21892±1463	2988±112
MCP$_1$	0.015 ±0.001	0.074 ±0.007	0.030 ±0.002	148±6	2422±82	336±19
MCP$_n$	0.016 ±0.001	0.131 ±0.012	**0.037** ±0.002	1285±91	22651±751	3008±145
MSP$_1$	0.013 ±0.001	0.099 ±0.007	0.024 ±0.001	156±7	2583±124	316±15
MSP$_n$	0.016 ±0.001	0.108 ±0.006	**0.037** ±0.004	1332±84	21447±1594	2931±203
MDPS$_1$	0.013 ±0.001	0.052 ±0.005	0.029 ±0.002	147±8	2620±79	313±15
MDPS$_n$	0.013 ±0.001	0.053 ±0.005	0.026 ±0.002	1283±81	22026±1599	2900±111
MCWPS	0.013 ±0.001	0.046 ±0.004	0.030 ±0.003	148±6	2612±151	321±20
MDWPS	0.008 ±0.002	0.058 ±0.004	0.018 ±0.002	139±4	2361±141	299±11

(for each chromosome, after each mutation attempt) needed to decide whether the mutation results should be retained or another mutation attempt should be used.

A greedy selection of the worst pure strategies (MCWPS and MDWPS) turned out to be an ineffective approach. This is most likely attributed to the fact that considering the quality of individual pure strategies in isolation from the other elements of a given mixed strategy may not be the right approach. Even if a single pure strategy is weak on its own, it may play a crucial role in the overall mixed strategy by rendering the decision that is unfavorable for the Defender to be also unprofitable for the Attacker.

Overall, the results show that repetition of mutation operation generally leads to improvement of SSGs outcomes, though at the expense of significant increase in computation time. Hence, in situations when computational cost is less important and obtaining the best possible result is critical, the proposed modifications offer a viable alternative to the base EASG formulation.

References

[1] Michalewicz, Z. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer Berlin Heidelberg, 1996.

[2] Neri, F. and Cotta, C. Memetic algorithms and memetic computing optimization: A literature review. *Swarm and Evolutionary Computation*, 2:1–14, 2012.

[3] Sinha, A., Fang, F., An, B., Kiekintveld, C., and Tambe, M. Stackelberg Security Games: Looking Beyond a Decade of Success. In *Proceedings of the 27th IJCAI conference*, pages 5494–5501. 2018.

[4] Breton, M., Alj, A., and Haurie, A. Sequential stackelberg equilibria in two-person games. *Journal of Optimization Theory and Applications*, 59(1):71–97, 1988.

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

[6] Żychowski, A. and Mańdziuk, J. Evolution of Strategies in Sequential Security Games. In *Proceedings of the 20th AAMAS conference*, pages 1434–1442. 2021.

[7] Karwowski, J., Mańdziuk, J., Żychowski, A., Grajek, F., and An, B. A memetic approach for sequential security games on a plane with moving targets. In *Proceedings of the 33rd AAAI conference*, volume 33, pages 970–977. 2019.

[8] Żychowski, A., Mańdziuk, J., Bondi, E., Venugopal, A., Tambe, M., and Ravindran, B. Evolutionary approach to Security Games with signaling. *Proceedings of the 31st IJCAI conference*, pages 620–627, 2022.

[9] Żychowski, A. and Mańdziuk, J. Learning attacker's bounded rationality model in security games. In *Proceedings of the 28th ICONIP*, pages 530–539. 2021.

[10] Karwowski, J., Mańdziuk, J., and Żychowski, A. Sequential stackelberg games with bounded rationality. *Applied Soft Computing*, 132:109846, 2023.

[11] Eiben, A. E., Michalewicz, Z., Schoenauer, M., and Smith, J. E. Parameter control in evolutionary algorithms. *Parameter setting in evolutionary algorithms*, pages 19–46, 2007.

[12] Karwowski, J. and Mańdziuk, J. A Monte Carlo Tree Search approach to finding efficient patrolling schemes on graphs. *European Journal of Operational Research*, 277:255–268, 2019.

[13] Bošanský, B. and Čermak, J. Sequence-form algorithm for computing stackelberg equilibria in extensive-form games. In *Proceedings of the 29th AAAI conference*, pages 805–811. 2015.

[14] Van Dijk, M., Juels, A., Oprea, A., and Rivest, R. L. Flipit: The game of "stealthy takeover". *Journal of Cryptology*, 26(4):655–713, 2013.