Evolutionary Approach to Security Games with Signaling

Adam Żychowski[1] , **Jacek Mańdziuk**[1*] , **Elizabeth Bondi**[2] , **Aravind Venugopal**[3] , **Milind Tambe**[2] and **Balaraman Ravindran**[3,4]
[1]Faculty of Mathematics and Information Science, Warsaw University of Technology
[2]Center for Research on Computation and Society, Harvard University
[3] Robert Bosch Centre for Data Science and AI, IIT Madras
[4]Department of Computer Science and Engineering, IIT Madras
{a.zychowski, mandziuk}@mini.pw.edu.pl, ebondi@g.harvard.edu,
aravindvenugopal19@gmail.com, milind_tambe@harvard.edu, ravi@cse.iitm.ac.in

Abstract

Green Security Games have become a popular way to model scenarios involving the protection of natural resources, such as wildlife. Sensors (e.g. drones equipped with cameras) have also begun to play a role in these scenarios by providing real-time information. Incorporating both human and sensor defender resources strategically is the subject of recent work on Security Games with Signaling (SGS). However, current methods to solve SGS do not scale well in terms of time or memory. We therefore propose a novel approach to SGS, which, for the first time in this domain, employs an Evolutionary Computation paradigm: EASGS. EASGS effectively searches the huge SGS solution space via suitable solution encoding in a chromosome and a specially-designed set of operators. The operators include three types of mutations, each focusing on a particular aspect of the SGS solution, optimized crossover and a local coverage improvement scheme (a memetic aspect of EASGS). We also introduce a new set of benchmark games, based on dense or locally-dense graphs that reflect real-world SGS settings. In the majority of 342 test game instances, EASGS outperforms state-of-the-art methods, including a reinforcement learning method, in terms of time scalability, nearly constant memory utilization, and quality of the returned defender's strategies (expected payoffs).

1 Introduction

Artificial intelligence is increasingly being used as a decision aide in many real-world scenarios, including for the protection of natural resources, such as wildlife, from illegal activities, such as wildlife poaching. These domains are often modeled as Green Security Games (GSGs) with two competing players [Fang *et al.*, 2015; Wang *et al.*, 2019], the defender and adversary. Finding the Stackelberg Equilibrium [Leitmann, 1978] in such games allows the defender to design an optimal strategy that minimizes attacks given limited resources. These strategies have proven to be effective in multiple real-world deployments, including in Queen Elizabeth National Park in Uganda [Yang *et al.*, 2014].

Recently, sensors have emerged for use in these domains as additional defender resources [de Cote *et al.*, 2013; Basilico and Gatti, 2014; Basilico *et al.*, 2015; De Nittis and Gatti, 2018; Bondi *et al.*, 2018]. Because they have the potential to detect an adversary in real time, they hold promise in improving GSG strategies. However, attacks may still occur if no human defender (henceforth, patroller) is nearby to interdict. Security games with signaling (SGS) have been proposed to help address this [Bondi *et al.*, 2020]. In these games, warning signals are sent to an adversary to convey that a patroller is responding, even sometimes when no patroller is truly responding. These must be used strategically [Bondi *et al.*, 2020; Xu *et al.*, 2018], as a total lack of response after a signal would render it meaningless.

However, these methods for solving SGS employ traditional optimization techniques, such as Linear Programming (LP) or Mixed Integer Linear Programming (MILP) and suffer from poor time scalability and large memory requirements. In this paper, we aim to solve SGS with Evolutionary Algorithms (EAs), which are inspired by the process of (biological) evolution. Recently, EAs have been successfully applied to sequential security games [Żychowski and Mańdziuk, 2021a], including games played in continuous space with moving targets [Karwowski *et al.*, 2019], or games with bounded rationality [Żychowski and Mańdziuk, 2021b].

Motivated by finding a scalable algorithm for SGS, we propose Evolutionary Algorithm for Security Games with Signaling (EASGS), the first metaheuristic for SGS based on an EA. In particular, we propose: (a) a novel defender-centered chromosome representation to facilitate an EA-based approach to SGS, (b) evolutionary operators, including a novel mutation and crossover approach, a population refreshing scheme, and local (memetic) optimizations, (c) a new set of benchmark games with various underlying graph structures, designed to reflect real-life SGS scenarios, (d) improved performance over state-of-the-art methods based on MILP and reinforcement learning (based on a substantial adaptation of [Venugopal *et al.*, 2021], which we call m-CombSGPO) in terms of expected payoffs, time scalability, and memory.

[*]Contact Author

Due to space limits, certain aspects related to the game scenario, EASGS implementation details, parameterization, and benchmark games are further discussed in supplementary material [Żychowski *et al.*, 2022].

2 Problem Definition

We consider repeated interactions between defender and adversary agents, assuming there is only one adversary. Note that EASGS can be easily applied to games with multiple adversaries, as only chromosome evaluation needs to be modified, however, we focus on one adversary for comparison purposes. The defender resources consist of l sensors and k patrollers. The sensors can detect an adversary in real time and notify the patrollers. There is uncertainty in the detection, as automatic detection in conservation drone imagery may be imperfect, particularly given occlusions such as trees. We specifically consider false negative detections as in [Bondi *et al.*, 2020; Basilico *et al.*, 2016; Basilico *et al.*, 2017].

In addition to sending notifications after a detection, sensors can also send two potential signals to an adversary, represented by σ_0 and σ_1. We consider σ_0 to be a weak signal, e.g., if considering a conservation drone as in [Bondi *et al.*, 2020], no lights are illuminated aboard the drone. On the other hand, σ_1 is a strong signal, e.g., lights are illuminated aboard the drone. Typically, we think of the strong signal as indicating that a patroller is (likely) responding, i.e., the lights turn on to deter the adversary. There is also observational uncertainty when the adversary observes the signal, again due to potential occlusions or difficulties viewing the true signal.

Sensors cannot directly prevent an attack, only signal to an adversary or notify the patrollers regarding an observation (which we assume is always done on an observation). If the adversary is deterred from attacking by a signal, both the defender and adversary receive 0 utility. If patrollers are at a target with an adversary, or nearby, they can interdict. The defender receives a positive utility if an attack is prevented successfully, while the defender receives a negative utility if an attack is successful. The opposite is true for the adversary.

The geography of the protected region is modeled using a graph, $G = (V, E)$, where each potential target in the region is represented by a vertex $\nu \in V$. If a sensor and patroller are at vertices connected by an edge $\epsilon \in E$, the patroller can successfully respond to that sensor's notification to prevent an attack. The number of vertices, $|V|$, is denoted by $\mathcal{N}$.

If there are no observations by either patrollers or sensors, patrollers may move to a neighboring vertex to have another chance of preventing an attack. This is known as the reaction stage, and is particularly useful if there is detection uncertainty, as a patroller would be able to move to a neighboring vertex with a sensor if the sensors have high uncertainty. Therefore, the order of the game is as follows: the defender first fixes their mixed strategy offline, then carries out a pure strategy allocation. The adversary proceeds to choose a target, and then may be detected by the defender's sensors. The defender may send a signal from the sensor and/or patrollers to respond to notifications, or re-allocate in the reaction stage. The adversary may observe the signal and react (flee). The game scenario is depicted in Figure 1.

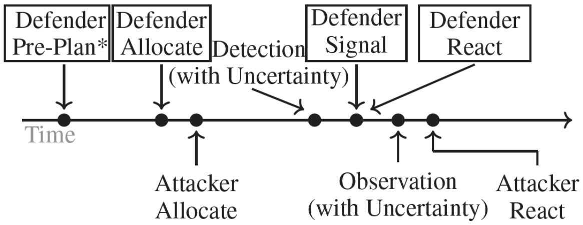


Figure 1: Players' actions timeline [Bondi *et al.*, 2020].

A defender pure strategy, e, therefore consists of an allocation of l sensors, an initial allocation of k patrollers, and a reaction stage allocation of k patrollers to neighboring vertices. The defender also has a signaling scheme, which is a series of probabilities over all targets and states. There are different signaling schemes when there is or is not a detection, as [Bondi *et al.*, 2020] found that it was necessary to occasionally signal without a detection due to uncertainty. Per the Stackelberg model, the adversary knows (only) the defender mixed strategy and signaling scheme.

In SGS, two sources of uncertainty are considered: *detection uncertainty* and *observational uncertainty*. The *detection uncertainty* is related to the limited capability of sensors, e.g., false negative detections. In real-life scenarios, a sensor's imperfection (whether the detection algorithm, the sensor itself, or both) can cause mistakes in reporting an adversary's presence. The most common situation is a false negative detection, i.e., a sensor misses the adversary due to, for example, poor visibility, weather conditions, occlusions, or even a power outage. In order to account for this, a probability of its occurrence is denoted by γ. Namely, the sensor with a probability equal to γ does not note the adversary's presence in the target during an attack.

The *observational uncertainty* is related to the adversary's observation of a signal. The true state may differ from the adversary's observation, i.e., the adversary may not notice the presence of a sensor or signal, possibly due to occlusions or other conditions. There are therefore 3 possible signaling states: (1) the sensor is not present in the target vertex at all (denoted by n), (2) the sensor is present, but does not send a strong signal (it sends a weak signal: σ_0), (3) the sensor is present and sends a strong signal: σ_1. Let us denote a conditional probability that the adversary will observe state $\hat{\omega}$ while the true signaling state is ω by $P[\hat{\omega}|\omega]$. Then, we can define an observational uncertainty matrix Π.

$$\Pi = \begin{bmatrix} P[n|n] & P[n|\sigma_0] & P[n|\sigma_1] \\ P[\sigma_0|n] & P[\sigma_0|\sigma_0] & P[\sigma_0|\sigma_1] \\ P[\sigma_1|n] & P[\sigma_1|\sigma_0] & P[\sigma_1|\sigma_1] \end{bmatrix}$$

3 Related Work

[Bondi *et al.*, 2020] achieves state-of-the-art performance solving the problem described in the previous section. We omit the details for the sake of brevity, but in short, they adapt the multiple linear programs (LPs) approach from [Conitzer and Sandholm, 2006]. By maximizing the defender expected utility for an LP at each vertex, the strategy yielding the highest defender expected utility of all LPs is the optimal strategy overall. Unfortunately, each LP is exponential in the number of possible pure strategies. The baseline method proposed in [Bondi *et al.*, 2020] solves each exponential LP to get an exact solution. We will also compare with this when possible, and refer to it as the exponential LP method (SELP).

In order to solve the problem more efficiently, [Bondi *et al.*, 2020] adapt the branch-and-price method. The overall idea is that they can (1) speed up solving each individual LP by using column generation, i.e., adding pure strategies gradually based on the result of a mixed integer linear program (MILP), and (2) use the branch and bound technique by solving a relaxation of each LP and pruning. We will denote this algorithm as branch-and-price (SBP) in comparisons. The most efficient method proposed in [Bondi *et al.*, 2020] uses the same branch-and-price method, but greedily generates an initial set of pure strategies for the LPs. We will denote this as branch-and-price with warm-up (SBP+W).

Another approach (called CombSGPO) proposed in [Venugopal *et al.*, 2021] is based on reinforcement learning, using competitive gradient descent and a multi-agent Deep Q-Network [Van Hasselt *et al.*, 2016] (MADQN) to solve a zero-sum game between the defender and the adversary on a grid graph. Our game model differs from [Venugopal *et al.*, 2021] by imposing no restrictions on the game graph topology, allowing a wider range of adversary's observation-reaction scenarios, and using different defender's strategy representation. We modify CombSGPO to work with our general-sum game model on any type of graph and call it m-CombSGPO (**m**odified CombSGPO). m-CombSGPO uses an actor-critic algorithm for computing the defender mixed strategy for allocation and reallocation, in the space of defender pure strategies [Bondi *et al.*, 2020]. Observing this mixed strategy, the adversary, represented by an actor-critic algorithm, chooses the target. Since we have a reaction stage instead of a patrolling stage like in [Venugopal *et al.*, 2021], instead of using a MADQN, we use a sensor signaling network, a neural network to choose the sensor (drone) signaling strategy, followed by an adversary decision neural network to choose whether the adversary should continue attacking or flee, both trained using the policy gradient algorithm [Sutton *et al.*, 2000]. m-CombSGPO is used as a reinforcement learning baseline.

4 Evolutionary Algorithm

We propose a novel Evolutionary Algorithm for Security Games with Signaling (EASGS). EASGS is a population-based algorithm, which means that a set of individuals representing solutions (i.e., defender's mixed strategies) iteratively evolve to find the optimal solution. Each individual is represented in the form of a chromosome. Initially, a population of n_{pop} chromosomes is randomly generated. Then, evolutionary operators (crossover, mutation, local optimization, and selection), which are described in detail in the following subsections, are repeatedly applied in subsequent generations. A total of n_{pop} chromosomes are maintained throughout the process of applying these operators, and ultimately, this leads to a final set of solutions.

Chromosome representation. Each chromosome encodes a valid solution, i.e., a candidate defender's mixed strategy, in the form of a list of pure strategies and their respective probabilities, and the sensors' signaling strategy:
$CH_j = \{(e_1^j, q_1^j), \ldots, (e_i^j, q_i^j), \ldots, (e_{d_j}^j, q_{d_j}^j), \boldsymbol{\Psi_j^\theta}, \boldsymbol{\Phi_j^\theta}\}$
where CH represents the chromosome, $j \in \{1, \ldots, n_{pop}\}$ is the individual's identifier, d_j is the number of pure strategies included in the mixed strategy represented by the chromosome CH_j, $e_i^j \in \mathcal{E}$ is a pure strategy in the set of pure strategies, $q_i^j \in [0, 1]$ is the probability of pure strategy e_i^j, $\sum_{i=1}^{d_j} q_i^j = 1$.
$\theta \in \{\bar{s}, s^+, s^-\}$ are allocation states: $\bar{s}$ when no patroller is in the sensor's neighbourhood (i.e., no one can respond to a notification), s^+ when a sensor has a patroller who will visit in the reaction stage (and could respond), s^- when no patroller will visit in the reaction stage (but there is a patroller in at least one neighbouring vertex who could respond),
$\boldsymbol{\Psi_j^\theta} = [\Psi_{j,1}^\theta, \Psi_{j,2}^\theta, \ldots, \Psi_{j,\mathcal{N}}^\theta]$ is the sensors' signaling strategy when an adversary is detected, i.e., $\Psi_{j,\nu}^\theta$ is the conditional probability[1] that the sensor at vertex ν sends signal σ_0 given the allocation state θ and that it detects an adversary,
$\boldsymbol{\Phi_j^\theta} = [\Phi_{j,1}^\theta, \Phi_{j,2}^\theta, \ldots, \Phi_{j,\mathcal{N}}^\theta]$ is the sensors' signaling strategy in the case of no adversary detection, i.e., $\Phi_{j,\nu}^\theta$ is the conditional probability that the sensor at vertex ν sends signal σ_0 given the allocation state θ and that it does *not* detect an adversary.

Pure strategies, e, are represented by tuples which contain defender's decisions according to the game rules described in problem definition section: $e = (V_p, V_s, V_r)$, where $V_p = \{\nu_1^p, \nu_2^p, \ldots, \nu_k^p\} \subseteq V$ is the patrollers' allocation plan, i.e., a set of vertices in which patrollers will be placed, $V_s = \{\nu_1^s, \nu_2^s, \ldots, \nu_l^s\} \subseteq V$ is the sensors' allocation plan, $V_r = \{\nu_1^r, \nu_2^r, \ldots, \nu_k^r\} \subseteq V$ is the patrollers' reallocation plan, i.e., the neighboring vertices to which each patroller moves if no adversaries are observed; the i_k-th patroller moves from vertex $\nu_{i_k}^p$ to $\nu_{i_k}^r$ (if relevant edge in the graph exists).

Observe that in [Bondi et al., 2020], a pure strategy is defined in a vertex-centered way, i.e. each vertex is assigned an allocation state (please consult [Bondi *et al.*, 2020] for the details). In EASGS, we propose a defender-centered strategy representation. Instead of focusing on the state of each vertex, a list of "actions" to be performed by the defender is encoded (i.e., to which vertices patrollers/sensors will be allocated/reallocated). Both encodings are equivalent, albeit, the latter one makes formulation of evolutionary operators (especially mutation) simpler and more effective.

Initial population. At the beginning, the population is initialized with individuals containing only one pure strategy: $\forall_j \, d_j = 1 \wedge q_1^j = 1$. These strategies are generated randomly according to the game rules defined in the problem definition, i.e., V_p, V_s, V_r are randomly selected subsets of V, and each element of $\boldsymbol{\Psi^\theta}$ and $\boldsymbol{\Phi^\theta}$ (signaling probability) is chosen uniformly at random from the unit interval.

[1] In [Bondi *et al.*, 2020], ψ^θ / x^θ, where x is the marginal probability of state θ.

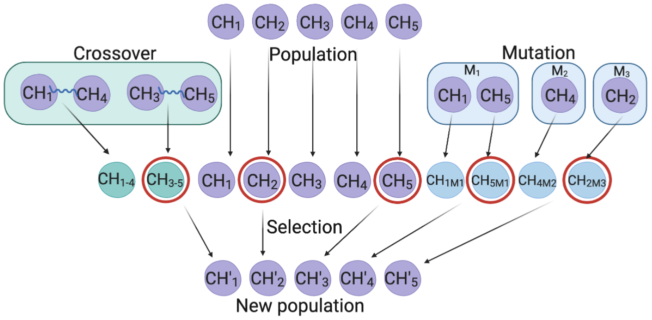


Figure 2: An example of EASGS operators application.

Crossover. The crossover operation combines two chromosomes by merging their sets of pure strategies such that the resulting child strategy contains a union of two parent strategies. A probability of each pure strategy in the resulting child chromosome depends on the previous probability (in the parent chromosome) and its utility (defender's payoff in case of playing this pure strategy). Formally, the probability of pure strategy e in the child chromosome is set to: $2^{u(e)}q$ where q is the probability of strategy e in the parent chromosome and $u(e)$ is its utility, i.e. the defender's payoff when playing strategy e, normalized to interval $[-1; 1]$. In particular, probability of the best pure strategy is doubled (its normalized utility equals 1), whereas that of the pure strategy with the lowest utility is halved (normalized utility equals -1). After that, the probabilities of all pure strategies are normalized so as to sum to 1. The intuition behind the above setting is to strengthen pure strategies with higher utilities and weaken lower-valued ones while, at the same time, not neglecting any of them. The above crossover definition extends the one proposed in [Żychowski and Mańdziuk, 2021a] in which utilities of strategies are not considered and all probabilities are first halved, and then normalized to 1.

Sensors' signaling strategies $\boldsymbol{\Psi^\theta}$ and $\boldsymbol{\Phi^\theta}$ in the child chromosome contain the averaged signaling probability from each parent. A result of the crossover operation on chromosomes CH_1 and CH_2 will be the new chromosome, $CH_{1\text{-}2}$:

$$\begin{aligned}&CH_{1\text{-}2} = \\ &\{(e^1_1, 2^{u(e^1_1)}q^1_1), (e^1_2, 2^{u(e^1_2)}q^1_2), \ldots, (e^1_{d_1}, 2^{u(e^1_{d_1})}q^1_{d_1}), \\ &(e^2_1, 2^{u(e^2_1)}q^2_1), (e^2_2, 2^{u(e^2_2)}q^2_2), \ldots, (e^2_{d_2}, 2^{u(e^2_{d_2})}q^2_{d_2}), \\ &\boldsymbol{\Psi^\theta_{1\text{-}2}}, \boldsymbol{\Phi^\theta_{1\text{-}2}}\},\end{aligned}$$

where
$\boldsymbol{\Psi^\theta_{1\text{-}2}} = [\frac{1}{2}(\Psi^\theta_{1,1} + \Psi^\theta_{2,1}), \ldots, \frac{1}{2}(\Psi^\theta_{1,\mathcal{N}} + \Psi^\theta_{2,\mathcal{N}})]$,
$\boldsymbol{\Phi^\theta_{1\text{-}2}} = [\frac{1}{2}(\Phi^\theta_{1,1} + \Phi^\theta_{2,1}), \ldots, \frac{1}{2}(\Phi^\theta_{1,\mathcal{N}} + \Phi^\theta_{2,\mathcal{N}})]$.

Repeating crossover operation in subsequent generations without any means of strategy removing would lead to chromosomes with a large number of pure strategies with tiny probabilities. To avoid this, after the crossover operation, each pure strategy e^j_i in the child chromosome CH_j is deleted with probability $(1 - q^j_i)^2$. The choice of deletion probability was empirically found to keep a good balance between the removal of less significant strategies (with lower q^j_i values) and the overall size of a mixed strategy. After deletion, the probabilities of the remaining pure strategies are normalized so as to sum to 1, i.e., $q^j_i\prime := \frac{q^j_i}{\Sigma_\tau q^j_\tau}$.

A set of chromosomes that are subject to crossover is chosen randomly in the following way. First, each individual from the population is selected for the crossover with crossover probability $\mathcal{P}_c$. Then, all selected chromosomes are randomly paired, and a crossover operation is performed. In the case of an odd number of individuals, a randomly chosen one is removed. From each set of parents, one new child chromosome is created and added to the population. An example crossover process is presented in Figure 2. The crossover operation helps in exploitation of the solution space by mixing strategies found so far.

Mutation. The mutation operator is applied to each chromosome CH_j from the current population independently with probability $\mathcal{P}_m$. There are 3 types of mutations: *probability change* (M_1), *allocation/signaling strategy modification* (M_2) and *coverage improvement* (M_3).

Probability change ($M_1 : \mathbb{R}^{d_j} \mapsto \mathbb{R}^{d_j}$) assigns a new probability ρ (uniformly drawn from the unit interval) to a randomly selected pure strategy e^j_i. Then, a softmax function is applied to all probabilities in a chromosome: $\forall_i \; q^j_i\prime := \frac{q^j_i}{\Sigma_\tau q^j_\tau}$. Thus, the M_1 mutation can be formulated as follows: $M_1([q^j_1, \ldots, q^j_i, \ldots, q^j_{d_j}]) = [q^j_1\prime, \ldots, \rho, \ldots, q^j_{d_j}\prime]$. M_1 modifies the influence of particular pure strategies on the result.

Allocation/signaling strategy modification ($M^1_2 : \mathcal{E} \mapsto \mathcal{E}$, $M^2_2 : [0,1]^{\mathcal{N}} \mapsto [0,1]^{\mathcal{N}}$) randomly selects one of the pure strategies (M^1_2) or signaling strategy (M^2_2) in the mutated chromosome. In case of M^1_2, a random perturbation is applied to a randomly chosen part of that pure strategy (V_p, V_s, or V_r). It means that the randomly selected sensor/patroller allocation/reallocation vertex ν_{old} is changed to another one (ν_{new}), selected randomly: $M^1_2((V_p, V_s, V_r)) = (V'_p, V'_s, V'_r)$ and $\exists!_z V_z \neq V'_z : z \in \{p, s, r\}$, $V_z = (\nu_1, \ldots, \nu_{old}, \ldots, \nu_k)$, $V'_z = (\nu_1, \ldots, \nu_{new}, \ldots, \nu_k)$.

In the case of M^2_2, a random element of the signaling probability strategy $\boldsymbol{\Psi^\theta}$ or $\boldsymbol{\Phi^\theta}$ ($\theta \in \{\bar{s}, s^+, s^-\}$) is changed to the *complementary* probability: $M^2_2(\boldsymbol{\Psi^\theta}) = \boldsymbol{\Psi^{\theta\prime}}$ and $\exists!_{z \in \{1,\ldots,\mathcal{N}\}}$: $\boldsymbol{\Psi^\theta} = [\Psi^\theta_1, \ldots, \Psi^\theta_z, \ldots, \Psi^\theta_{\mathcal{N}}]$, $\boldsymbol{\Psi^{\theta\prime}} = [\Psi^\theta_1, \ldots, 1 - \Psi^\theta_z \ldots, \Psi^\theta_{\mathcal{N}}]$, with an analogous operation for $\boldsymbol{\Phi^\theta}$. By applying a random perturbation, M_2 aims to explore new areas of the search space to find potential improvements.

Coverage improvement ($M_3 : \mathcal{E} \mapsto \mathcal{E}$) is based on information from the evaluation process from the previous generation. In the evaluation procedure, the adversary's optimal strategy is computed (see Evaluation below), which contains a target vertex, ν_{adv}, that the adversary will choose. Thus, a natural idea is to increase the coverage probability of this vertex. This is realized by randomly finding a pure strategy that does not cover ν_{adv} (i.e., no patroller or sensor is allocated to this vertex), and adding ν_{adv} either to V_p or V_s in this strategy, in place of another randomly selected vertex ν_{rand}: $M_3((V_p, V_s, V_r)) = (V'_p, V'_s, V_r)$ and $\exists!_z V_z \neq V'_z : z \in \{p, s\}$, $V_z = (\nu_1, \ldots, \nu_{rand}, \ldots, \nu_k)$,

$V_z' = (\nu_1, \dots, \nu_{adv}, \dots, \nu_k)$. Observe that it is not guaranteed that this greedy approach will improve the defender's strategy. In some situations, however, the lack of target coverage is done purposefully and increases the final expected payoff.

If a chromosome is chosen for mutation based on $\mathcal{P}_m$, one of the three above described mutation types is randomly chosen and applied to the chromosome (see Figure 2). This procedure is repeated until a better-fitted chromosome is obtained or m_{limit} trials is reached. After each unsuccessful mutation attempt, the chromosome is reverted to its state before mutation. The last mutation attempt is kept even if it leads to a lower-fitted chromosome. In preliminary experiments we observed that, in case none of the mutation attempts returned a better individual, keeping the best mutated individual is not statistically superior to simply using the last resulting individual.

Evaluation. After population initialization, as well as crossover and mutation, a post-processing step (local optimization) takes place to ensure chromosomes' feasibility. Next, the evaluation procedure calculates the fitness value of each chromosome which is the defender's expected payoff when the mixed strategy encoded in the chromosome is played. Since it is proven in [Conitzer and Sandholm, 2006] that for Stackelberg Games there always exists at least one best adversary's response to the defender's mixed strategy in the form of a pure strategy, it is sufficient to evaluate the defender's strategy against all pure adversary strategies.

To this end, we first determine the adversary's strategy by iterating over all possible adversary's pure strategies. Each adversary's pure strategy contains a commitment to allocation (vertex to be attacked) and signaling reaction scheme (response to observed signals - run away or continue the attack). For each possible pair (a vertex and a signaling reaction scheme), players' payoffs are computed. Next, we choose the adversary's strategy yielding the best adversary payoff, and determine the defender's payoff for this scenario, which is considered the expected defender's payoff. This value is assigned as the chromosome fitness value.

To compute players' payoffs, we determine the marginal probability x_ν^θ of the presence of a patroller, a sensor, or no defender resource at the currently attacked vertex based on the list of allocation pure strategies e and their corresponding probabilities q. Then, for each defender's pure strategy, each of the three possible states ((1)-(3) below) is considered separately and assumed to be the final defender's strategy. The respective defender's payoff is multiplied by the marginal probability. The following three states are considered: (1) In the case of a patroller's presence in the vertex, the defender simply catches the adversary. (2) If neither a patroller nor a sensor are located in the attacked vertex, then the resulting payoff depends on the patroller's reallocation. If a patroller is reallocated to the attacked vertex in the reaction stage, the adversary will be caught. (3) The last possibility is that a sensor is located in the attacked vertex (and there is no patroller allocated). In that case, two scenarios must be considered, depending on whether or not the sensor detects the adversary. The probability of each scenario is defined by γ, the detection uncertainty. The probability of sending signal σ_0, given by $\boldsymbol{\Psi^\theta}$ and $\boldsymbol{\Phi^\theta}$, depends on several factors: the adversary's detection, the signaling strategy, and the reallocation state (whether there is a patroller in the neighborhood that will visit this vertex in the reaction stage). All the above information leads to computing the probability of a sensor's signaling strategy[2]:

$$\mathcal{P}_{\sigma_1}(\nu) = \gamma \sum_\theta (x_\nu^\theta \Phi_\nu^\theta) + (1-\gamma) \sum_\theta (x_\nu^\theta \Psi_\nu^\theta)$$

$$\mathcal{P}_{\sigma_0}(\nu) = \gamma \sum_\theta x_\nu^\theta (1 - \Phi_\nu^\theta) + (1-\gamma) \sum_\theta x_\nu^\theta (1 - \Psi_\nu^\theta)$$

where $\mathcal{P}_{\sigma_1}(\nu)$ and $\mathcal{P}_{\sigma_0}(\nu)$ are probabilities of sending signals σ_1 and σ_0 in vertex ν, respectively,
$\theta \in \{\bar{s}, s^+, s^-\}$ are allocation states,
x_ν^θ represents the marginal probability that vertex ν is in the allocation state θ.
x_ν^θ is computed based on allocation strategies (e_i, q_i), for example: $x_\nu^{s^+} = \sum q_i : \nu \in V_r \subset e_i$.

The next step is to determine the adversary's reaction. Based on the signaling strategy and probabilities of how certain signals can be observed by the adversary (Π), the adversary's reaction is computed which, together with the defender's strategy, allows for computing the final payoffs.

The detailed diagram of the evaluation procedure is presented in supplementary material.

Selection. In order to construct a population for the next generation, the selection procedure is executed on a pool of all parent individuals (the current population), and the results of crossover and mutation operators (offspring individuals). In the beginning of the selection process, a fixed number, n_e of *elite* individuals from this pool are automatically promoted to the new population. The elite individuals are top-ranked chromosomes with the highest fitness values.

Next, the following iterative procedure is executed until the next generation population is filled with n_{pop} individuals. In each iteration, two chromosomes are randomly drawn with return from the pool. Then, from this pair, the one with a higher fitness value is promoted to the next generation with probability $\mathcal{P}_{sp}$ (selection pressure). Otherwise, the lower-fitted one is promoted.

Stop condition. Multiple applications of the proposed operators can, in principle, lead to any arbitrary solution. Repeated crossover can create a mixed strategy with an arbitrary number of pure strategies, M_1 can set any probability distribution of these strategies, and M_2 applied multiple times is able to transform each arbitrary pure strategy to any other. Hence, any mixed strategy can be potentially achieved through repeated application of these operators, independently of the initial population selection. However, it is difficult to say anything about theoretical time requirements of the method. Providing theoretical convergence guarantees for EAs is generally a challenging task and formal results in this area occur rarely in the literature. Thus, the whole routine (crossover, mutation, local optimization, evaluation, and

[2] chromosome's identifier (j) is omitted for clarity in equations

selection) is repeated until some fixed number of generations n_{gen} is reached.

Furthermore, if for n_{ref} (refresh) generations a payoff of the top-ranked individual does not change, some fresh chromosomes are added to the population, i.e., we randomly select half of the population, which is replaced by the same number of new individuals with randomly assigned pure strategies, generated in the same way as in the initial population. The best individual is preserved and cannot be deleted. This approach avoids getting stuck in sub-optimal solutions and shifts the population to new potentially better areas.

5 Experimental Setup

Access to the data from the real-world scenario is strictly secured, e.g. to protect the locations of wild animals and rangers resources, so we randomly generated 342 artificial games with various utilities, graph architectures, numbers of vertices, patrollers, and sensors. All generated games are publicly available on github.com/easgs/benchmark_games.

EASGS was compared with the exact approach, SELP, and the three heuristic methods: SBP, SBP+W and m-CombSGPO, summarized in Related Work. Results for non-deterministic algorithms (including EASGS) were obtained in 30 independent runs per game instance. Tests were performed on a cluster running CentOS Linux 7 (Core) with Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.1 GHz with 128 GB RAM and 4 cores. EASGS source code can be found on github.com/easgs/source_code.

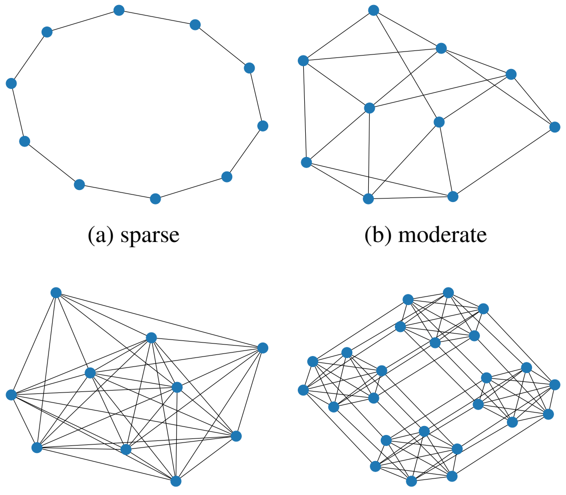
Benchmark games. Let $\mathcal{N}$ and a_d denote the graph size and the average vertex degree. Games were generated within four groups: *sparse* (50 games), with $a_d = 2$, *moderate* (50 games) ($a_d = \mathcal{N}/2$), *dense* (50 games) ($a_d = \mathcal{N} - 2$) and *locally-dense* (192 games), a set of connected cliques.

Sparse, moderate, and dense game graphs were random Watts-Strogatz graphs with $\mathcal{N} = 10, 20, \ldots, 100$. In *locally-dense* graphs, the number of cliques and their size varied from 3 to 10. *locally-dense* games are inspired by certain real-life scenarios, in which there exist areas (cliques) which patrollers can quickly explore, and these are connected via larger links, e.g., roads, which speed up travel. In [Xu *et al.*, 2020], patrollers travel to areas via motorbikes and then patrol on foot. The graphs are further described in supplementary material. Figure 3 presents four examples.

EASGS parameterization. EASGS parameters were tuned on a set of 12 games with 20 vertices (3 games of each type: *sparse*, *moderate*, *dense* and *locally-dense*), which were separated from the 342 EASGS benchmark graphs and not used during the method evaluation. Based on 5000 runs, the following parameter values were finally chosen: population size $n_{pop} = 200$, crossover probability $\mathcal{P}_c = 0.5$, mutation probability $\mathcal{P}_m = 0.8$, mutation repetition limit $m_{limit} = 10$, number of elite chromosomes $n_e = 2$, selection pressure $\mathcal{P}_{sp} = 0.8$, generations limit $n_{gen} = 2000$, number of generations between refreshes $n_{ref} = 300$.

6 Results and Discussion

The results are averaged over 30 runs and presented in 3 perspectives: payoffs, time scalability, and memory.



(a) sparse

(b) moderate

(c) dense

(d) locally dense

Figure 3: Sample benchmark game graphs.

Payoffs. SELP managed to calculate the payoffs only for games with up to 20 vertices. For larger games, its memory consumption exceeded the limit of 128 GB. Thus, we were able to compare obtained results with the optimal solution only for 30 game instances - 6 for each of *sparse*, *moderate* and *dense* type, and 12 *locally-dense* ones. Results for these games are presented in Table 1.

	SBP			SBP+W			EASGS		
sparse	**0.00**	**0.00**	**6/6**	**0.00**	**0.00**	**6/6**	0.90	0.01	1/6
moderate	15.77	0.17	1/6	7.92	0.09	**2/6**	**1.51**	**0.02**	1/6
dense	18.93	0.26	**0/6**	9.25	0.09	**0/6**	**1.98**	**0.03**	**0/6**
locally-dense	13.45	0.28	1/12	3.16	0.05	**4/12**	**1.42**	**0.02**	3/12

Table 1: In order to make a thorough comparison, we compare the best EASGS, SBP, SBP+W solutions with the optimal ones in terms of both the average difference (left columns) and the ratio between the defender's payoffs (middle columns). Right columns present fractions of games for which optimal solutions were found. Each method is assumed to have reached the optimal result if the returned payoff difference smaller than 0.01. The best results are bolded.

SBP and SBP+W reached optimal solutions in all *sparse* test games. However, for denser graphs, their accuracy degrades significantly, and overall, the leading method is EASGS. At the same time, EASGS reaches the exact optimal result less frequently (5 out of 30 games). In other words, EASGS is able to find areas with high quality solutions, but may have some difficulties with precise exploitation of those areas. m-CombSGPO (which is based on reinforcement learning) is not a suitable method for finding exact solutions. It has not reached the optimal solution for any game instance and therefore is omitted in Table 1.

Table 2 shows a comparison of heuristic methods based on all 342 test games. The results confirm that EASGS works better for all types of games except for *sparse* ones, whereas both SBP and SBP+W degrade on denser graphs. The reason may lie in the increasing branching factor of denser games, as

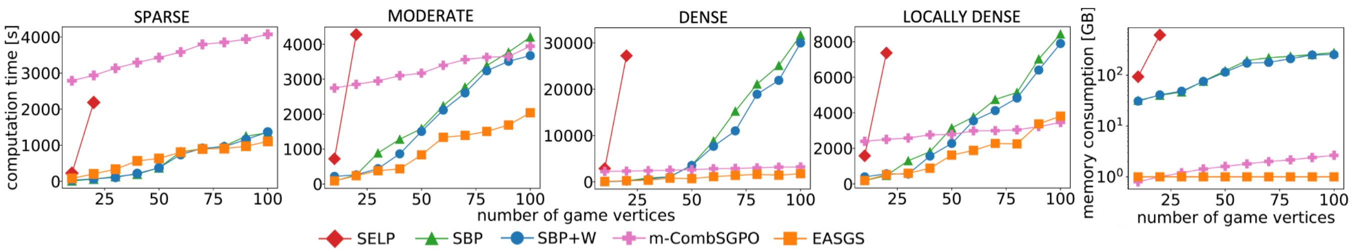


Figure 4: Time (left) and memory (right) scalability averaged over 5 games (m-CombSGPO had to be trained separately for each game).

	SBP	SBP+W	m-CombSGPO	EASGS
sparse	-86.68 (84%)	**-86.01 (92%)**	-419.86 (0%)	-91.32 (6%)
moderate	-75.01 (2%)	-72.75 (36%)	-255.73 (0%)	**-69.92 (62%)**
dense	-58.72 (2%)	-57.98 (34%)	-149.14 (0%)	**-51.47 (64%)**
locally-dense	-60.68 (4%)	-57.80 (26%)	-340.65 (0%)	**-54.36 (70%)**

Table 2: Comparison of the average defender's payoff across all benchmark games. In brackets are the percentages of games for which a given method obtained the winning result. Note that multiple methods could achieve the same winning result, so the scores may not sum to 100%. The best results are bolded.

each additional connection between vertices opens the possibility for patroller reallocation and enlarges the game strategy tree. In all cases m-CombSGPO reaches the lowest payoffs. Overall, for 200 out of 342 test game instances, EASGS yielded the best result. For each type of benchmark game, the advantage of the winning method (SBP+W for *sparse* games and EASGS for all other types) is statistically significant according to a 1-tailed paired t-test with a significance level equal to 0.05, and with a normal distribution of data checked by a Shapiro-Wilk test. However, when all 342 games of all types are considered together, it is not statistically significant, with the significance level equal to 0.08. EASGS is also a stable method. The average standard deviation (30 runs of each game) equals 0.86 (about 1.2% of the average absolute payoff - cf. Table 2) with the maximal value of 1.63.

Time scalability. Another critical aspect of the proposed algorithm is time scalability. Figure 4 compares time efficiency of the considered methods. Except for the smallest game graphs and moderate-size *sparse* ones (where SBP+W performs slightly better), EASGS outperforms all competitive methods, sometimes by a large margin (e.g. on *dense* graphs). The explanation of this phenomenon is related to the way the EASGS operations are implemented. Action complexity in EASGS is independent of the underlying graph density: mutation assigns vertices to patrollers/sensors randomly, and the topology of connections between vertices (and the number of them) do not influence those choices.

Memory consumption. Although memory consumption is typically not a key factor in recent years, in some cases (especially for methods based on traditional optimization techniques), memory utilization can hinder real-life applications. SELP, for example, consumes more than 200 GB of memory for small games with 20 vertices. At the other extreme, EASGS memory utilization is almost constant. The only data stored in EASGS memory is the game definition and the EA population. This is reflected in Figure 4, which presents memory utilization for all tested methods. Even for 100-vertex games, EASGS memory does not exceed 150 MB.

Qualitative analysis. The majority of EASGS resultant mixed strategies contained between 4 to 8 pure strategies (with the mean of 5.3). There wasn't a single resultant strategy with an uncovered vertex, i.e. for each vertex there was at least one pure strategy with a sensor or patroller assigned to that vertex, meaning the target coverage generated by EASGS was complete. Moreover, the patrollers were generally assigned to higher degree vertices, which allowed patrollers to respond to more targets (checked by sensors) during the reallocation stage. Targets with higher defender penalties were generally covered with higher probability. EASGS also considered adversary utilities by protecting targets of high value for the adversary, as well as graph architecture by assigning the vast majority of the sensors to vertices that are connected to targets with assigned patrollers.

7 Conclusion

The paper introduces a novel approach to solving Security Games with Signaling based on the Evolutionary Computation paradigm. The method maintains a population of potential solutions (mixed strategies) and modifies them using specially-designed operators which address domain characteristics. A comprehensive experimental evaluation confirmed that EASGS provides better payoffs than state-of-the-art methods, especially for *locally-dense* graphs that are inspired by real-world settings. Additionally, EASGS largely outperforms competitive methods in terms of time scalability. Thanks to nearly constant memory scalability and the ability to return a valid solution at any time during the execution process, EASGS can be employed to larger games that are beyond the capacity of the state-of-the-art methods. Being well-suited for field computing requirements, EASGS takes us a step closer towards deploying real-world applications on a larger scale to protect valuable natural resources and biodiversity. As future work, we plan to test how well EASGS performs in the field by integrating it with more advanced sensors based on real-world object detection from images.

Acknowledgements

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

Research was sponsored by the Army Research Office and was accomplished under Grant Number: W911NF-17-1-0370. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

References

[Basilico and Gatti, 2014] Nicola Basilico and Nicola Gatti. Strategic guard placement for optimal response to alarms in security games. In *Proceedings of the 13th AAMAS*, pages 1481–1482, 2014.

[Basilico *et al.*, 2015] Nicola Basilico, Giuseppe De Nittis, and Nicola Gatti. A security game model for environment protection in the presence of an alarm system. In *International Conference on Decision and Game Theory for Security*, pages 192–207, 2015.

[Basilico *et al.*, 2016] Nicola Basilico, Giuseppe De Nittis, and Nicola Gatti. A security game combining patrolling and alarm-triggered responses under spatial and detection uncertainties. In *Proceedings of the 30th AAAI*, pages 397–403, 2016.

[Basilico *et al.*, 2017] Nicola Basilico, Giuseppe De Nittis, and Nicola Gatti. Adversarial patrolling with spatially uncertain alarm signals. *Artificial Intelligence*, pages 220–257, 2017.

[Bondi *et al.*, 2018] Elizabeth Bondi, Fei Fang, Mark Hamilton, Debarun Kar, Donnabell Dmello, Jongmoo Choi, Robert Hannaford, Arvind Iyer, Lucas Joppa, Milind Tambe, et al. Spot poachers in action: Augmenting conservation drones with automatic detection in near real time. In *Proceedings of the 32nd AAAI*, pages 7741–7746, 2018.

[Bondi *et al.*, 2020] Elizabeth Bondi, Hoon Oh, Haifeng Xu, Fei Fang, Bistra Dilkina, and Milind Tambe. To signal or not to signal: Exploiting uncertain real-time information in signaling games for security and sustainability. In *Proceedings of the 34th AAAI*, pages 1369–1377, 2020.

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

[de Cote *et al.*, 2013] Enrique de Cote, Ruben Stranders, Nicola Basilico, Nicola Gatti, and Nick Jennings. Introducing alarms in adversarial patrolling games. In *Proceedings of the 12th AAMAS*, pages 1275–1276, 2013.

[De Nittis and Gatti, 2018] Giuseppe De Nittis and Nicola Gatti. Facing multiple attacks in adversarial patrolling games with alarmed targets. *arXiv preprint arXiv:1806.07111*, 2018.

[Fang *et al.*, 2015] Fei Fang, Peter Stone, and Milind Tambe. When security games go green: Designing defender strategies to prevent poaching and illegal fishing. In *Proceedings of the 24th IJCAI*, pages 2589–2595, 2015.

[Karwowski *et al.*, 2019] Jan Karwowski, Jacek Mańdziuk, Adam Żychowski, Filip Grajek, and Bo An. A memetic approach for sequential security games on a plane with moving targets. In *Proceedings of the 33rd AAAI*, pages 970–977, 2019.

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

[Sutton *et al.*, 2000] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in NIPS*, pages 1057–1063, 2000.

[Van Hasselt *et al.*, 2016] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI*, pages 2094–2100, 2016.

[Venugopal *et al.*, 2021] Aravind Venugopal, Elizabeth Bondi, Harshavardhan Kamarthi, Keval Dholakia, Balaraman Ravindran, and Milind Tambe. Reinforcement learning for unified allocation and patrolling in signaling games with uncertainty. In *Proceedings of the 20th AAMAS*, pages 1353–1361, 2021.

[Wang *et al.*, 2019] Yufei Wang, Zheyuan Ryan Shi, Lantao Yu, Yi Wu, Rohit Singh, Lucas Joppa, and Fei Fang. Deep reinforcement learning for green security games with real-time information. In *Proceedings of the 33rd AAAI*, pages 1401–1408, 2019.

[Xu *et al.*, 2018] Haifeng Xu, Kai Wang, Phebe Vayanos, and Milind Tambe. Strategic coordination of human patrollers and mobile sensors with signaling for security games. In *Proceedings of the 32nd AAAI*, pages 1290–1297, 2018.

[Xu *et al.*, 2020] Lily Xu, Shahrzad Gholami, Sara Mc Carthy, Bistra Dilkina, Andrew Plumptre, Milind Tambe, et al. Stay ahead of poachers: Illegal wildlife poaching prediction and patrol planning under uncertainty with field test evaluations. In *36th IEEE Conference on Data Engineering*, pages 1898–1901, 2020.

[Yang *et al.*, 2014] Rong Yang, Benjamin Ford, Milind Tambe, and Andrew Lemieux. Adaptive resource allocation for wildlife protection against illegal poachers. In *Proceedings of the 13th AAMAS*, pages 453–460, 2014.

[Żychowski and Mańdziuk, 2021a] Adam Żychowski and Jacek Mańdziuk. Evolution of strategies in sequential security games. In *Proceedings of the 20th AAMAS*, pages 1434–1442, 2021.

[Żychowski and Mańdziuk, 2021b] Adam Żychowski and Jacek Mańdziuk. Learning attacker's bounded rationality model in security games. In *International Conference on Neural Information Processing*, volume 1516, pages 530–539, 2021.

[Żychowski *et al.*, 2022] Adam Żychowski, Jacek Mańdziuk, Elizabeth Bondi, Aravind Venugopal, Milind Tambe, and Balaraman Ravindran. Evolutionary approach to Security Games with signaling. *arXiv preprint arXiv:2204.14173*, 2022.