# A Memetic Approach for Sequential Security Games on a Plane with Moving Targets

**Jan Karwowski,**[1] **Jacek Mańdziuk,**[1] **Adam Żychowski,**[1] **Filip Grajek,**[1] **Bo An**[2]
[1] Warsaw University of Technology, Faculty of Mathematics and Information Science, Koszykowa 75, 00-662 Warsaw, Poland
[2] Nanyang Technlogical University School of Computer Science and Engineering, Singapore 639798
{j.karwowski, j.mandziuk, a.zychowski}@mini.pw.edu.pl, boan@ntu.edu.sg

**Abstract**

This paper introduces a new type of Security Games (SG) played on a plane with targets moving along predefined straight line trajectories and its respective Mixed Integer Linear Programming (MILP) formulation. Three approaches for solving the game are proposed and experimentally evaluated: application of an MILP solver to finding exact solutions for small-size games, MILP-based extension of recently published zero-sum SG approach to the case of general-sum games for finding approximate solutions of medium-size games, and the use of Memetic Algorithm (MA) for medium-size and large-size game instances, which are beyond MILP's scalability. Utilization of MA is, to the best of our knowledge, a new idea in the field of SG. The novelty of proposed solution lies specifically in efficient chromosome-based game encoding and dedicated local improvement heuristics. In vast majority of test cases with known equilibrium profiles, the method leads to *optimal solutions* with *high stability* and *approximately linear time scalability*. Another advantage is an iteration-based construction of the system, which makes the approach essentially an *anytime method*. This property is of paramount importance in case of restrictive time limits, which could hinder the possibility of calculating an exact solution. On a general note, we believe that MA-based methods may offer a viable alternative to MILP solvers for complex games that require application of approximate solving methods.

## 1 Introduction

Security Games (SG) are a research area focusing on applying tools that originated in game theory to the task of building patrol strategies and schedules for security forces in certain real-life situations, modeled as games, specifically the Stackelberg Game model (Leitmann 1978). SG research encompasses broad spectrum of game settings (e.g. (Xu et al. 2017; Guo et al. 2016; Gholami et al. 2016; Karwowski and Mańdziuk 2016; Cermak et al. 2016; Gan, An, and Vorobeychik 2015)) and has been applied in various domains (An, Tambe, and Sinha 2017).

While majority of SG papers address discrete game spaces (Dickerson et al. 2010; Gutierrez, Juett, and Kiekintveld 2013), an important property of our model is continuous space of target and defender locations, which can be used to model a class of real-life protection situations involving passenger or cargo ships cruising according to a fixed schedule. Two methods for solving the game are proposed. The first one utilizes an MILP formulation and is solved by professional solver. In order to make the game tractable by the solver, a game discretization procedure, which transforms a continuous strategy space game into its discretized form, equivalent in terms of Stackelberg Equilibrium (SE) strategies, is proposed. This discretization procedure together with specific looseless game compression are applied prior to using the solver.

Due to limited scalability of the MILP approach, an alternative metaheuristic algorithm for solving larger game instances is proposed. The method relies on a Memetic Algorithm (MA) with specifically designed chromosome based game representation and efficient local optimization heuristic, and is applied *directly to continuous strategy space games*, with no need for game discretization.

**Related Work.** While the mainstream SG research considers static target settings there are several recent papers addressing the moving target scenarios:

- MRMT$_{sg}$ and CASS approaches (Fang, Jiang, and Tambe 2013), which model continuous Defender's strategy (similar to our problem), but in 1-dimensional space. Their solution considers only Markovian strategies for the Defender.
- Games played on a graph where targets and defenders move in discrete space of graph nodes (Bosansky et al. 2011). That work also considers only Markovian policies for the Defender.
- A model called Security Games on a Plane (Gan et al. 2017) is the closest to ours, with a major difference of being a single step game. Our approach directly extends the game model proposed in (Gan et al. 2017) by considering sequential game formulation.
- In (Wang et al. 2018) a game model similar to our discretized game version was considered and solved using compact game representation, game abstraction and constraint-generation approach. The method requires the game to be zero-sum and additionally simplifies calculation of independent events to a sum of probabilities in a Linear Program (LP), what may lead to overoptimistic results if the probability of catching is high.

All the above approaches utilize model-specific properties to make the search in exponential Defender's decision space feasible. A more general framework - *column generation* - was proposed in (Jain et al. 2010). Its application requires an efficient heuristic for selection of the Defender's move sequences whose inclusion in mixed strategy would improve the result. A reasonable heuristic would rely on reduced cost values in LP that calculates the equilibrium strategy. Such an approach, however, would work only for an LP formulation that does not contain integer variables, what prevents its effective application to our problem. In order to remove integer variables from the program, application of branching is needed, which in our case would yield a tree with an exponential number of nodes, thus making Jain et al. (2010) approach ineffective, even with pricing-based pruning.

**Our Contribution.** While each of the above-mentioned papers have some commonalities with our game model, none of them is close enough to allow application of the respective solution method to find *non-Markovian* strategy for *continuous*, *sequential* game studied in this paper. The main contribution of this work is

- A new approach to Security Games that relies on Memetic Algorithms. To the best of our knowledge this is the first example of application of MA (or generally Evolutionary Algorithms) in SG domain. The method scales *approximately linearly* within the tested ranges of game steps, targets and defenders and offers *high stability* of obtained solutions. Encouraging results of its application to our game model make us believe that, on a general note, MA-based methods may offer a viable alternative to MILP solvers in the case of complex games that require application of approximate solving methods.

Furthermore,

- A new sequential game model for SG with moving targets, played on a plane is introduced as an extension of (Fang, Jiang, and Tambe 2013) and (Gan et al. 2017). A game abstraction procedure and corresponding MILP formulation for calculation of exact results are proposed.
- A modification of the Wang et al. (2018) method that allows its application to our game model (which is non-zero-sum) is proposed and experimentally evaluated.

## 2 Game description

A moving targets scenario considered in this paper is motivated by a real-life sea transportation situation of a tourist harbor in the Mediterranean Sea in Southern Europe. In this setting, there are a number of $n_f$ ferries (the *targets*) which carry people and commodities, and some number $n_d$ of fast patrolling boats (defenders) securing these transports. The patrolling boats are governed by a Central Security Unit (the *Defender*). A ferry can be potentially threatened by a fast-speed pirate boat (the *Attacker*). Each ferry $F_j$ cruises back and forth between the assigned pair of terminals following a predefined schedule $S_j$ which is defined as a list of departure times from each of these two terminals (alternately). Ferries move with a constant speed and follow a straight line route. At each moment a ferry is protected *iff* there is at least one defending unit located within a protection radius $r$.

| game parameter | notation |
|---|---|
| number of game steps | $m$ |
| number of defenders | $n_d$ |
| number of targets | $n_f$ |
| number of terminals | $n_b$ |
| time step length | $\tau$ |
| game duration | $T = m\tau$ |
| locations of terminals | $b_i = (x_i, y_i), i = 1, \ldots, n_b$ |
| targets (with 4 payoffs and a schedule) | $F_j = (U_j^{a+}, U_j^{a-}, U_j^{d+}, U_j^{d-}, S_j)$, $j = 1, \ldots, n_f$ |
| defenders | $D_k, k = 1, \ldots, n_d$ |
| defender maximum speed | $v_d^{max}$ |
| target speed | $v_f$ |
| protection radius | $r$ |
| attack length | $\tau_A = \lambda\tau$ for some integer $\lambda$ |

Table 1: Game parameters with their respective notation. Positions of targets and defenders at time $t$ are denoted by $F_j(t), D_k(t)$ for $j = 1, \ldots, n_f,\ k = 1, \ldots, n_d,\ t = 0, \ldots, (m-1)\tau$.

**Game initial position and game steps.** Each defender is assigned a starting point in one of the terminals. Each target starts from one of the two terminals which define both ends of their repetitive two-way itinerary. The game is divided into $m$ time steps of equal length ($\tau$) and the beginning of each step marks a decision point, i.e. subsequent decision points $t_1, \ldots, t_m$ occur at times $0, \tau, 2\tau, \ldots, (m-1)\tau$, for the $m$-step game of length $m\tau$.

**Defender's deterministic strategy.** In each discrete time point the Defender decides about moving their units to new positions which must fulfill the feasibility constraints, i.e. each Defender's unit cannot move farther than $\tau \cdot v_d^{max}$, where $v_d^{max}$ is a speed limit for defending units (same for all of them).

The Attacker's decision in each time point is either to choose the target to be attacked or wait. In the former case the attack lasts for a predefined time $\tau_A$ (same for all targets). The Attacker can decide to perform at most one attack during the entire game.

**Game scoring and results.** Four payoffs are assigned to each target $F_j$: $U_j^{d+}$ is a reward for the Defender in case of catching the Attacker, $U_j^{a-}$ is a penalty for the Attacker in case of being caught, $U_j^{d-}$ is a penalty for the Defender in case the successful attack takes place, and $U_j^{a+}$ is a reward for the Attacker for a successful attack realization. Table 1 presents all game parameters and their respective notation. A game solution is not a single deterministic strategy (a sequence of moves of the defending units), but a *mixed strategy*, i.e. a probability distribution of deterministic strategies. Each deterministic strategy profile represents a certain game scenario with particular outcomes assigned to each of the players. The mixed strategy with the highest expected payoff for the Defender constitutes the game solution. We model the game as a Stackelberg game where the Defender commits to a mixed strategy and then, based on its analysis, the Attacker chooses the best response strategy, breaking ties in favor of the Defender.

An outcome of a given deterministic strategy profile (particular game playout) is calculated as follows. Suppose the

**Algorithm 1:** Compressed game calculation

```
Π_d^c ← ∅, gridPositions ← reasonableGridPositions();
allUnitPositions ← gridPositions^{n_d};
Function PossibleCompactMoves(currentPositions,
 protectionSequence)
    if length(protectionSequence) = m then
        Π_d^c ← Π_d^c ∪ {protectionSequence}
    else
        forall (protectionVector, positions)←
          equivalenceClasses(protectionVector,
          possibleSuccessors(currentPositions,
          allUnitsPositions)) do
            PossibleCompactMoves(positions,
              protectionSequence⊕protectionVector)
        end
    end
PossibleCompactMoves(startPosition, emptyList)
```

Attacker decides to attack target $F_j$ at decision point $p$ (time $p\tau$). The attack is successful *iff* target $F_j$ is not protected during the entire attack timespan, i.e. the interval $[p\tau, (p+\lambda)\tau]$. Please recall, that target $F_j$ is protected at a given time moment *iff* there exists a defending unit $D_k$ for which $dist(D_k(t), F_j(t)) \leq r$, where $dist(\cdot,\cdot)$ is Euclidean distance on a plane. Once the attack attempt is finished the game ends and the expected payoffs are calculated depending on the choice of target $F_j$ and the outcomes of the attack (success or failure). If the Attacker refrains from attacking any target during the entire game, both players' payoffs are equal to 0.

## 3 An exact MILP approach

Exact solutions for the game instances are derived using the DOBSS method (Paruchuri et al. 2008) which calculates exact Stackelberg Equilibrium strategy using MILP. To apply the MILP method we need first to discretize the game to make it feasible for the MILP solver, and then apply a procedure for significant reduction of game (state and move) space, to reduce the number of variables and constraints. Once the equilibrium strategy for this compact game is calculated using MILP solver, it is converted back into equivalent strategy for the original (non-compact) discrete game.

### Discrete game transformation

MILP representation requires that the game has finite strategy space for both players and therefore cannot be applied directly to the original continuous game described in Section 2. A method of transferring continuous game into its equivalent discrete form relies on *grid step* parameter ($d$) responsible for the smoothness of the discretization. It can be easily seen that for values of $d$ small enough the process yields a game equivalent to the original continuous game in terms of equilibrium strategies under the assumption that coordinates of targets and defenders positions, in all game steps, are rational.

Continuous space is divided into a discrete grid of points $G$. The $(0,0)$ coordinate of $G$ is located in the arbitrary chosen terminal. A step size $d$ is a function of the defender's protection range $r$ and the maximum distance $l$ that a defender can move during one time step ($l = \tau v_d^{max}$): This way, each defending unit assures continued protection, in a sense that a center of their protection area in the next time step remains within the range of the previous protection area. Formula $d = \min\{r, l/2\}$ yields a good balance between coarse granularity of space (and thus faster computation) and flexibility of potential strategies. The component $l/2$ guarantees that a move is technically possible to be executed even for large $r$. The grid range is limited by positions of the farthest terminals.

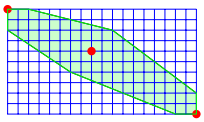


Figure 1: A set of nodes (within a green polygon) which are potentially interesting from the Defender's viewpoint. Red dots denote terminal locations.

In the initial state, the defending units are placed in the nodes closest to their respective starting terminals. Their moves are restricted to the grid points. In each turn, a unit can move to any of the nodes for which the Euclidean distance is $\leq l$. The targets move according to their schedules and their positions are calculated in continuous space.

### Compact discrete game

While it would be possible to apply the exact MILP directly to the above discretized game, the Defender's strategy space of this game is extremely large $\approx (\pi(l/d)^2)^{n_d m}$. Therefore, the game is further transformed into alternative, smaller decision space, which is equivalent in terms of equilibrium strategies to a discretized game. In this compact game representation, the Defender's deterministic strategy consists in a direct choice of the targets which are protected in each round. The main difficulty in this process is deciding which protection sequences are possible under physical game constraints (moving speed and range of defenders and targets). Let's denote by $\Pi_d^g$ – the set of Defender's deterministic strategies in the original (grid based) discrete game, by $\mathcal{F}$ – the set of all targets, by $\mathcal{P} := (2^{\mathcal{F}})^m$ – the finite set of all potential target protection sequences, by $cov : \Pi_d^g \to \mathcal{P}$ – a function that provides a protection sequence produced by a given Defender's strategy, and by $\Pi_d^c := \{\pi_d^c | (\exists \pi_d^g \in \Pi_d^g) cov(\pi_d^g) = \pi_d^c\}$ – the set of Defender's deterministic strategies in the compact game. Definition of $\Pi_d^c$ ensures that a strategy in the compact game is possible *iff* there exists a corresponding Defender's strategy in the original discrete game space, so there is equilibrium equivalence between the compact game and the original game. The Attacker strategy space remains unchanged. A pseudocode of the $\Pi_d^c$ calculation method is presented in Algorithm 1. A procedure reasonableGridPositions in Algorithm 1 calculates the relevant subset of all grid nodes, removing from $\Pi_d^g$ the nodes that are redundant for producing protection vectors. First, a convex hull of all terminals is cal-

culated. Then, all points that are within the hull or in a distance less than $r+1$ from the hull compose the set of relevant nodes. A simple example situation is presented in Figure 1, where a subset of green nodes out of all grid (blue) nodes is selected. Three red points denote terminal locations. The process of conversion from the original discrete game to the compact game representation is computationally expensive (exponential in the number of defending units and targets count) but in return allows the resulting MILP formulation (for compact game) to fit in memory and consequently optimal strategies for these games can be derived.

### An MILP formulation

MILP game formulation is adapted from DOBSS (Paruchuri et al. 2008) that was used for calculating optimal strategies. The only difference is the lack of a dimension representing different types of adversaries in our formulation (denoted by index $l$ in (Paruchuri et al. 2008)) as our game in not Bayesian SSG and there is only one (type of) Attacker. The resulting program has the following form:

$$\begin{array}{rl} & \max_{q,z,a} \sum_{i\in X}\sum_{j\in Q} U_{ij} z_{ij} \\ \text{s.t.} & \sum_{i\in X}\sum_{j\in Q} z_{ij} = 1 \\ (\forall_{i\in X}) & \sum_{j\in Q} z_{i,j} \le 1 \\ (\forall_{j\in Q}) & q_j \le \sum_{i\in X} z_{ij} \le 1 \\ & \sum_{j\in Q} q_j = 1 \\ (\forall_{j\in Q}) & 0 \le (a - \sum_{i\in X} C_{ij}(\sum_{h\in Q} z_{ih})) \\ (\forall_{j\in Q}) & (a - \sum_{i\in X} C_{ij}(\sum_{h\in Q} z_{ih})) \le (1-q_j)M \\ & z_{ij}\in[0,1], \quad q_j \in \{0,1\}, \quad a \in \mathbb{R}, \end{array} \tag{1}$$

where $X := \Pi_d^c$ is a set of possible Defender's target protection sequences, $Q$ — a set of possible Attacker's moves, $U_{ij}$ — the Defender's payoff when moves $i$ and $j$ are played, $C_{ij}$ — the respective Attacker's payoff. Variables $z_{ij}$ denote probabilities that the Attacker and the Defender play pure strategies $j$ and $i$, resp. Note, that $z_{ij}$ will be non-zero only for one pure Attacker strategy $j^*$ being an optimal Attacker's response strategy. $q_j$ and $a$ are auxiliary variables used to enforce optimal Attacker's response.

## 4 A modification of SMOS approach for general-sum games

In this section we propose a modification to LP solution of Stackelberg Model of Oil Siphoning problem (SMOS) presented in (Wang et al. 2018) that allows its application to our *general-sum* game model. Moreover, since the Attacker's strategy space in our model is relatively small (there are $1 + mn_f$ possible moves) the constraint-generation part of the original solution is not utilized in our approach. The third major aspect of the Wang et al. (2018) approach — a grid abstraction — is applied without modifications.

The rest of this section presents our modification of LP suitable for non-zero-sum games. We use notation from (Wang et al. 2018) - which differs from the notation used in the rest of our paper - for the sake of ease in tracing the changes introduced to the original model. A modified MILP is described by eqs. (2)–(16) and extends the Wang et al. (2018) approach by having $|F|$ times more real variables and $O(|F|^2)$ integer variables (not present in the base formulation).

$$\max \sum\nolimits_f U^d(f,f) \tag{2}$$

$$0 \le a - \sum\nolimits_{g\in F} U^a(g,f) \le (1-q_f)\cdot M \quad \forall f \in F \tag{3}$$

$$\sum\nolimits_{f\in F} q_f = 1, \tag{4}$$

$$\begin{aligned} c(i,t_k,f) = \sum\nolimits_{j\in N(i)} fl_{((i,t_k),(j,t_{k+1}),f)} \\ \forall i \in Z, k \in 0,\dots\tau-1, f\in F \end{aligned} \tag{5}$$

$$\begin{aligned} c(i,t_k,f) = \sum\nolimits_{j\in N(i)} fl_{((j,t_{k-1}),(i,t_k),f)} \\ \forall i \in Z, k \in 1,\dots\tau, f\in F \end{aligned} \tag{6}$$

$$q_f \cdot m = \sum\nolimits_{i\in Z} c(i,t_k,f) \quad k\in\{0,\tau\}, f\in F \tag{7}$$

$$q_f \cdot DS(i) = c(i,0,f) \quad \forall i \in S, f\in F \tag{8}$$

$$1 - m_1(g,f)M_2 \le dpp(g,f) \le 1 \quad \forall f,g\in F \tag{9}$$

$$\begin{aligned} \sum\nolimits_{(i,t)\in R(f)} c(i,t,g) - m_2(g,f)M_2 \le \\ dpp(g,f) \le \sum\nolimits_{(i,t)\in R(f)} c(i,t,g) \quad \forall f,g\in F \end{aligned} \tag{10}$$

$$m_1(g,f) + m_2(g,f) = 1, \quad \forall f,g\in F \tag{11}$$

$$q_g - m_3(g,f)M_2 \le dpp_{neg}(g,f) \le q_g \forall f,g\in F \tag{12}$$

$$\begin{aligned} 1 - dpp(g,f) - m_3(g,f)M_2 \le dpp_{neg}(g,f) \\ \le 1 - dpp(g,f) \forall f,g\in F \end{aligned} \tag{13}$$

$$m_3(g,f) + m_4(g,f) = 1, \quad \forall f,g\in F \tag{14}$$

$$\forall i,g,f m_i(g,f) \in \{0,1\} \quad \forall f q_f \in \{0,1\} \tag{15}$$

$$\begin{aligned} \forall i,j,k,f,g fl_{((i,t_k),(j,t_{k+1}),f)} \in \mathbb{R}, dpp(g,f)\in\mathbb{R} \\ \forall i,k,f c(i,t_k,f) \in \mathbb{R} \quad \forall g,f dpp_{neg}(g,f)\in\mathbb{R} \end{aligned} \tag{16}$$

where payoff calculation functions are:

$$U^a(g,f) = dpp_{neg}(g,f)U^{a+}(f) + dpp(g,f)U^{a-}(f) \tag{17}$$

$$U^d(g,f) = dpp_{neg}(g,f)U^{d-}(f) + dpp(g,f)U^{d+}(f) \tag{18}$$

In order to make the Wang et al. (2018) solution applicable to non-zero-sum games, a dedicated set of variables describing defenders' coverage flow for each Attacker's strategy was introduced. Index $f$ in variables $fl$ and $c$ denotes the Attacker's sequence and plays analogous role to index $j$ in variables $z$ in DOBBS (cf. eq. (1)). A set of variables $q_f$ and eq. (4) ensure the existence of only one active Attacker's strategy, and eq. (3) ensures that this active Attacker's strategy is the best Attacker's response. Eqs. (7) and (8) guarantee that among flow variables $c$ only those with index $f$ of currently active Attacker are non-zero. The latter equation additionally ensures proper defenders' starting points. Eqs. (9)–(11) force the $dpp$ values to be equal to $\min\{1, \sum c(i,t,g)\}$ for the respective sums of coverages. Please note that, contrary to the original work, general-sum formulation requires auxiliary integer variables $m_1, m_2$ to ensure reaching the minimum, as variables $dpp$ are utilized not only in the objective function, but also in constraint (3). The need of these variables is the reason that prevents application of the column generation approach mentioned in section 1. Due to eqs. (12)—(14), $dpp_{neg}$ variables have value of $1 - dpp$ for the active strategy ($g$) and 0 for the remaining

Attacker's strategies. Eq. (15) explicitly lists integer variables added to the program (original formulation does not use integer variables). The remaining (not described) equations (2), (5), (6) and (16) are adapted directly from (Wang et al. 2018). For the current defenders' coverage, functions $U^a(g, f)$ and $U^d(g, f)$ defined in (17) and (18) return payoffs of the respective players for an Attacker's strategy $f$ *iff* $g$ is the active Attacker's strategy. Otherwise, they return 0.

## 5 Memetic approach

The approximate approach proposed in this paper employs the MA solution scheme with problem dependent local in-generation optimization heurstic. MA (Ong, Lim, and Chen 2010; Chen et al. 2011; Neri, Cotta, and Moscato 2012) enhances population-based Evolutionary Algorithm (EA) by means of adding a distinctive local optimization phase. The underlying idea of memetics is to use local optimization techniques or domain knowledge to improve potential solutions between consecutive EA generations. A synergetic combination of a local improvement scheme with evolutionary operators leads to complex and powerful solving methods which are applicable to a wide range of problems (Neri and Cotta 2012). In the remainder of this section, all components of the proposed MA approach are discussed in more detail. In particular, a novel problem encoding and efficient local improvement heuristic.

**Chromosomes.** Each chromosome $CH_q, q = 1, \ldots, p\_size$ in a population encodes mixed Defender's strategy by defining positions of all their units in all decision points of a game. A simple strategy ($SS$) is encoded as a set of lists: $SS = \{(D_1(t_1), \ldots, D_1(t_m)), \ldots, (D_{n_d}(t_1), \ldots, D_{n_d}(t_m))\}$. Chromosomes are of various lengths, i.e. may contain different numbers of SSs. Each SS in a chromosome is assigned a probability of occurrence. These probabilities sum up to 1 within each chromosome. Thus, a chromosome is a set of pairs:

$$CH_q = \{(SS_1^q, p_1^q), \ldots, (SS_{l_q}^q, p_{l_q}^q)\}, \ \sum_{i=1}^{l_q} p_i^q = 1 \quad (19)$$

where $l_q$ is length of the $q$-th chromosome $CH_q$, $p_i^q$, $i \in \{1, \ldots, l_q\}$ is probability of occurrence of strategy $SS_i^q$.

**Fitness function.** A fitness of a chromosome is equal to the expected Defender's payoff obtained when playing a strategy encoded by that chromosome. In SG, the Defender's payoff is calculated under the assumption of the optimal Attacker's strategy, i.e. the optimal Attacker's decision regarding the possible attack (either optimal choice of the target and attack time or resignation from attacking any target). This decision relies on the probability distribution of defending units positions, which - by definition of SG - are known to the Attacker. To this end, all possible attack scenarios are calculated and the one with the highest Attacker's expected payoff (if positive) is chosen, or a decision to refrain from attacking is assumed (with the Attacker's payoff equal to 0). Let's denote by $rank$ the respective Defender's payoff, by $U^x(F_j, t_i), x \in \{d, a\}, j \in \{1, \ldots, n_f\}, i \in \{1, \ldots, m\}$ the Defender's ($x = d$) and the Attacker's ($x = a$) payoffs, resp. in the case of attacking target $F_j$ at step $t_i$, and let $(F_*, t^*) = \arg\max_{F_j, t_i} U^a(F_j, t_i)$. If $U^a(F_*, t^*) \geq 0$ then $rank := U^d(F_*, t^*)$, otherwise $rank := 0$ (the Attacker would not attempt).

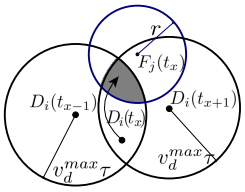



Figure 2: Changing position $D_i(t_x)$ during local optimization. New position is uniformly chosen from the gray area. It depends on target position in step $t_x$ ($F_j(t_x)$) and positions of unit $D_i$ in the neighboring steps ($D_i(t_{x-1})$, $D_i(t_{x+1})$).

Calculation of possible attack scenarios in each decision point additionally allows to break ties between chromosomes with the same fitness in the selection process. Namely, if two (or more) chromosomes have the same fitness value, the sum of $m$ Defender's payoffs (one per each decision point) is computed under the assumption that the Attacker would decide to attack in that decision point, regardless of his payoff. In other words, in this auxiliary fitness (*in-rank grade*), it is assumed that in each decision point the Attacker chooses the most rewarding (for them) target to attack and the respective Defender's payoffs are summed up across all $m$ steps: *in-rank grade* $:= \sum_{i=0}^{m} U^d(F_*, t_i)$, where $F_* = \arg\max_{F_j} U^a(F_j, t_i), j \in \{1, \ldots, n_f\}$. If $U^a(F_*, t_i) < 0$ then $U^d(F_*, t_i) := 0$. The greater the *in-rank grade* the higher the preference for that chromosome when tied with other chromosomes in the $rank$ fitness measure. This auxiliary fitness measure is very effective in situations when the resulting strategy provides targets protection in some decision points but not in all of them. In such cases, a natural assumption that back-ups the reasonability of the auxiliary fitness measure is that strategies with fewer decision points of low Defender's payoffs are preferable.

**Initial population.** In each chromosome, initial positions $(D_1(t_1), D_2(t_1), \ldots, D_{n_d}(t_1))$ of defending units are defined in randomly selected terminals. In the next steps, given a position in decision point $t_{x-1}$, a position in point $t_x$ (for $x \in \{2, \ldots, m\}$) is chosen uniformly among all positions reachable by the defending unit (restricted by the speed limit $v_{max}^d$) in which it secures at least one target (a distance to that target is $\leq r$). If there are no such positions, any reachable position is randomly selected.

**Mutation.** Each chromosome is mutated with *mutation rate* probability. A single mutation changes defending unit position in uniformly chosen time point $t_x, x \in \{2, \ldots, m-1\}$. New position must be feasible, i.e. is chosen uniformly from the common parts of the circles with centers in $D_i(t_{x-1})$, $D_i(t_{x+1})$ and radius equal to $v_d^{max}\tau$. Mutation operation is repeated *mutation repeats* times for each chromosome which undergoes mutation.

**Crossover.** Crossover operation combines two chromosomes by merging their sets of simple strategies. A probability of each SS in the resulting child chromosome is equal to

the half of the respective probability in the parent chromosome. The result of this operation on chromosomes $CH_{q_1}$ and $CH_{q_2}$ will be the new chromosome $CH_{q_{12}}$:

$$CH_{q_{12}} = \{(SS_1^{q_1}, p_1^{q_1}/2, (SS_2^{q_1}, p_2^{q_1}/2), \ldots, (SS_{l_{q_1}}^{q_1}, p_{l_{q_1}}^{q_1}/2), (SS_1^{q_2}, p_1^{q_2}/2), (SS_2^{q_2}, p_2^{q_2}/2), \ldots, (SS_{l_{q_2}}^{q_2}, p_{l_{q_2}}^{q_2}/2)\}$$

To avoid creating mixed strategies with too many simple strategies after crossover operation each $SS_i^q$ may be deleted with probability $1 - p_i^q$. After this deletion process, the sum of $p_i^q$ of the removed strategies is distributed among all remaining SSs, proportionally to their probabilities. New chromosome is added to the existing population.

**Local optimization (LO).** Memetic optimization takes place after genetic operators (mutation and crossover), but before selection. It attempts to improve solution by optimizing iteratively the defending units positions in all decision points. The new position in step $t_x$ is restricted by the feasibility of the overall encoded solution, i.e. depends on positions in steps $t_{x-1}$ and $t_{x+1}$ and the speed limit $v_{max}^d$. Among all feasible positions the algorithm first selects the ones which secure the greatest number of targets, and then chooses one of them at random. This memetic optimization is iteratively repeated for each defender in each SS within each chromosome. Figure 2 visualizes the area of all possible positions after performing local optimization in step $t_x$, assuming one target $F_j$.

LO was identified as a critical part of MA. Defender's scores obtained in 3000 preliminary tests performed with standard Genetic Algorithm (GA) approach (i.e. MA without LO phase) were lower by $0.38$ in average, and in none of the cases the score of GA excelled that of MA. Thanks to the LO phase, Defender's units are more concentrated on the targets, i.e. without this phase their positions are more scattered.

**Selection.** First, two best-fitted chromosomes (elite) are copied to the next generation. Then, the following tournament selection takes place. Two chromosomes are randomly chosen from the population. With probability equal to *selection pressure* the one with the higher fitness is added to the next generation. Otherwise, the lower-fitted chromosome is promoted. The above routine is repeated until the next generation contains *population size* chromosomes.

**Parameter tuning.** Tuning of parameters was constrained to *a priori* selected arbitrary set of values for each of them. Each parameter was separately tuned on 20 randomly defined games, with all other parameters set to their default values. The best value for each parameter was chosen based on the obtained average Defender's payoff. Table 2 presents the sets of tested parameter values as well as their default and finally chosen settings.

## 6 Results and discussion

The methods were tested on $6150$ games generated with the following values of the main parameters:

- $m \in \{2, 4, 6, 8, 10\}$
- $n_f \in \{1, 2, 3, 4, 5, 6, 8, 10, 15, 20, 25, 30, 35, 40, 45, 50\}$
- density $\rho \in \{0.2, 0.1, 0.05\}$, which determines the side length $a$ of (squared) game area, i.e. $a := \frac{\sqrt{n_f}}{\rho}$
- $n_d \in \{max(1, n_f - 2), \ldots, n_f + 1\}$

For each selected parameter set $(m, n_f, n_d, \rho)$ 10 games were uniformly generated with the following ranges of the remaining parameters:

- $n_b$ - integer from interval $[2\sqrt{n_f}; 3\sqrt{n_f}]$
- $U_j^{d+}, U_j^{a+} \in [0, 1], \qquad U_j^{d-}, U_j^{a-} \in [-1, 0]$
- $\tau_A = \tau, \qquad v_f = v_d^{max} = 1.0, \qquad r = 1.0$
- terminals coordinates $\in [0, a]^2$

In each game, for each target, a pair of associated terminals was uniformly selected and the cruising schedule was defined with uniformly selected waiting times in each terminal $\in [0, 3\tau]$. Starting positions of defending units were assigned to randomly selected terminals. MILP problems were solved using Gurobi solver in version 8.0 (Gurobi Optimization, Inc. 2018). Modification of (Wang et al. 2018) was calculated using the same solver, with three different values of game abstraction (grid aggregation) parameter $s = 1, 2, 4$. In the case of MA 10 trails were run for each game.

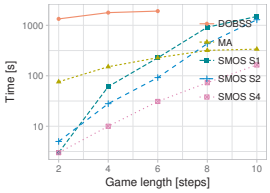
| parameter | value |
|---|---|
| population size | $10, 20, 50, 100, \underline{200}, 500, 1000, \mathbf{2000}$ |
| # generations | $10, \underline{100}, 200, 500, 1000, \mathbf{2000}$ |
| mutation rate | $0.0, \mathbf{0.2}, 0.4, \underline{0.6}, 0.8, 1$ |
| mutation repeats | $1, 5, \mathbf{10}, 25, \underline{50}, 100, 200$ |
| crossover rate | $0.0, 0.2, 0.4, 0.6, \underline{0.8}, \mathbf{0.9}, 1$ |
| selection pressure | $0.7, 0.8, \underline{\mathbf{0.9}}, 1.0$ |
| # elite chromosomes | $\underline{\mathbf{2}}$ |

Table 2: Parameter setting of the memetic approach. Default values used during tuning procedure are underlined. Best values are bolded.

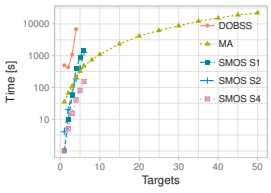
**Optimality.** Only $605$ games were tractable for DOBSS on available hardware (Intel Xeon 4116 with 256GB RAM) due to time and memory scaling problems. These $605$ games contained up to $4$ steps, 3 targets and 2 defender's units. Out of these $605$ games for which the optimal solution was computed by DOBSS, MA yielded the same as DOBSS, optimal Defender's payoff in $599$ cases, with the error margin set to $10^{-4}$. In the case of Wang et al. (2018) method, the number of solved games depended on $s$ and was highest for $s = 4$, in which case $3146$ instances were solved. Due to simplifying assumption about summing up probabilities of dependent events and utilization of grid abstraction, the method, similarly to MA, yields approximate SE results. Figure 6 presents histograms of differences in results between SMOS method for various $s$ values and MA. Negative ranges denote the advantage of MA, which can be observed for all values of $s$, most notably $s = 4$.

**Repeatability.** MA turned out to be a very stable method. The average standard deviation of $6150$ games (each run $10$ times) was equal to $3 \cdot 10^{-3}$ and in only $554$ ($9\%$) instances had a value greater than $10^{-5}$.

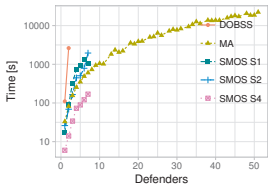
**Scalability.** Figures 3 and 4 present computation time for considered methods with respect to the main game parame-



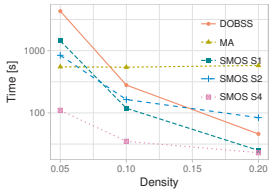

(a) Time scalability w.r.t. the number of game steps.




(b) Time scalability w.r.t. the number of targets.




(c) Time scalability w.r.t. the number of defenders.




(d) Time scalability w.r.t. density parameter.

Figure 3: Scalability results.

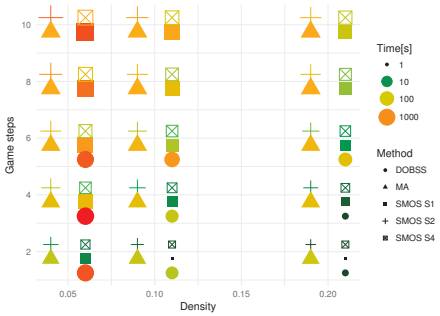



Figure 4: Scalability results w.r.t. density and the number of game steps.

ters. Increasing the number of targets and defenders (Figures 3b and 3c) causes exponential time growth for both SMOS and DOBSS approaches. With respect to MA only SMOS, $s = 4$ is faster, albeit only until $n_f = 7$ and $n_d = 9$, and at the cost of significantly worse results (cf. Figure 6).

Analysis of the number of game steps (Fig. 3a) leads to a conclusion that DOBSS runtime is significantly greater than that of MA, but its growth with respect to increasing game length is not as steep as in reference to the increasing number of targets (Fig. 3b) or Defender's units (Fig. 3c). SMOS solution for short games runs significantly faster, but does not scale well for larger $m$ values. For $m \geq 8$ MA gradually outperforms SMOS, $s = 1, 2$ and for $m = 10$ becomes time-comparable with SMOS, $s = 4$.

Results presented in Figures 3d and 4 show that MA, which operates in continuous space, is insensitive to density parameter $\rho$, while significantly longer computational times are obtained by solver based methods for smaller $\rho$ values. The reason for that is game discretization and compressing method where the search space is limited by a convex hull which, by definition, contains the number of grid points inversely proportional to $\rho^2$. Consequently, the resulting games are more complex with targets located farther apart from each other (in grid-based distance measure).

**Additional experiments.** In order to further assess scalability of MA with respect to game length additional tests were performed for this method with $m \in \{15, 20, 25, 30\}$. The remaining approaches were capable of solving only small subsets of games for these higher values of $m$, so their outcomes are not presented. Good scalability of MA was confirmed also for larger games, as presented in Figure 5.

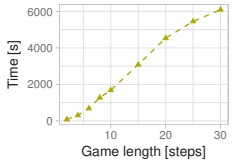



Figure 5: Scalability of MA for $m > 10$.

In summary, the MA method scales significantly better than solver based approaches. The range of solvable games (using moderate computational resources) is significantly broader, with the ability to solve 30-step games with greater numbers of targets and defending units. Certainly, these extended MA capabilities when compared to DOBSS come at a price of lacking rigorous guarantee of solution optimality, however, a direct comparison with the outcomes within the DOBSS feasible range of game parameters indicates that MA solutions are of very high quality — optimal in vast majority of the cases.

## 7 Conclusions and future work

In this paper, an extension of the Security Games on a Plane model introduced in (Gan et al. 2017), which relies on adding time dimension, is proposed and solved. In the resulting game model, targets are moving back and forth along straight line trajectories according to fixed schedules and the defending units are freely moving on the whole 2-dimensional plane with certain speed limit. Each defender covers all targets located within the circular area of predefined radius around its current position. The proposed game model fits a wide class of real-life situations of cruising targets protection, e.g. passenger ferries or cargo boats.

For solving the game, a suitable DOBSS-based formulation is proposed. Due to exponential memory requirements with respect to game length, games solvable by DOBSS are relatively small – of 6 steps, at most. This inherent DOBSS limitation prevents finding exact solutions beyond a certain limit of game complexity.

Furthermore, an extension to non-zero-sum games of the recently published method (Wang et al. 2018) is proposed and experimentally evaluated as a viable alternative to DOBSS formulation. While this modified version of Wang et al.'s (2018) method allows to solve significantly larger game instances, it still suffers from scalability issues, beyond certain limits.

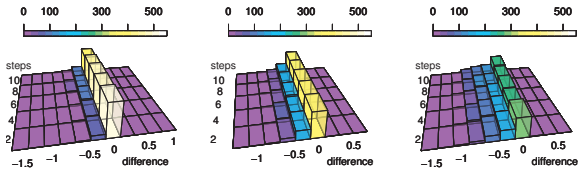



Figure 6: Differences between Defender's utility values obtained with MA and SMOS for various settings of $s$.

For this reason, besides the solver-based solutions, an approximate solution method relying on the Memetic Algorithm is designed. Even though the MA approach cannot guarantee finding optimal Defender's strategies (likewise the Wang et al. (2018) method), experimental results indicate that, in vast majority of the cases, it generates optimal strategies - the same as those provided by the MILP solver. Except for *high accuracy* of results the main asset of the MA method is *approximately linear time scalability* and *high stability* of solutions for mid-size and large game instances.

It is also worth to underline that the MA method is flexible and can be easily adjusted to other SG models. Another advantage is its iteration-based construction, which makes the approach essentially an *anytime method*, what seems to be a crucial property in case of restrictive time limits, which could hinder the possibility of calculating the exact solution.

Considering the above properties of the proposed MA method we believe that is presents a suitable approach for solving SG with Moving Targets (and possibly other SG formulations) for larger game instances whose complexity prevents application of solver based methods.

**Acknowledgments.** This work was supported by the National Science Centre, grant number 2017/25/B/ST6/02061.

## References

An, B.; Tambe, M.; and Sinha, A. 2017. Stackelberg security games (SSG): Basics and application overview. In *Improving Homeland Security Decisions*. Cambridge University Press. 485–507.

Bosansky, B.; Lisy, V.; Jakob, M.; and Pechoucek, M. 2011. Computing time-dependent policies for patrolling games with mobile targets. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, 989–996.

Cermak, J.; Bosansky, B.; Durkota, K.; Lisy, V.; and Kiekintveld, C. 2016. Using correlated strategies for computing Stackelberg Equilibria in extensive-form games. In *Thirtieth AAAI Conference on Artificial Intelligence*, 439–445.

Chen, X. S.; Ong, Y. S.; Lim, M. H.; and Tan, K. C. 2011. A multi-facet survey on memetic computation. *IEEE Transactions on Evolutionary Computation* 15(5):591–607.

Dickerson, J. P.; Simari, G. I.; Subrahmanian, V.; and Kraus, S. 2010. A graph-theoretic approach to protect static and moving targets from adversaries. In *International Conference on Autonomous Agents and Multiagent Systems*, 299–306.

Fang, F.; Jiang, A. X.; and Tambe, M. 2013. Optimal patrol strategy for protecting moving targets with multiple mobile resources. In *International Conference on Autonomous Agents and Multiagent Systems*, 957–964.

Gan, J.; An, B.; and Vorobeychik, Y. 2015. Security games with protection externalities. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 914–920.

Gan, J.; An, B.; Vorobeychik, Y.; and Gauch, B. 2017. Security games on a plane. In *AAAI Conference on Artificial Intelligence*, 530–536.

Gholami, S.; Wilder, B.; Brown, M.; Thomas, D.; Sintov, N.; and Tambe, M. 2016. Divide to defend: Collusive security games. In *International Conference on Decision and Game Theory for Security*, 272–293. Springer.

Guo, Q.; An, B.; Vorobeychik, Y.; Tran-Thanh, L.; Gan, J.; and Miao, C. 2016. Coalitional security games. In *International Conference on Autonomous Agents and Multiagent Systems*, 159–167.

Gurobi Optimization, Inc. 2018. Gurobi optimizer reference manual. `http://www.gurobi.com/documentation/8.0/`.

Gutierrez, E.; Juett, J.; and Kiekintveld, C. 2013. Generating effective patrol strategies to enhance U.S. border security. *Journal of Strategic Security* 6(3):152—-159.

Jain, M.; Kardes, E.; Kiekintveld, C.; Ordónez, F.; and Tambe, M. 2010. Security games with arbitrary schedules: A branch and price approach. In *AAAI Conference on Artificial Intelligence*, 177–190.

Karwowski, J., and Mańdziuk, J. 2016. Mixed strategy extraction from UCT tree in security games. In *European Conference on Artificial Intelligence*. IOS Press. 1746–1747.

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

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

Neri, F.; Cotta, C.; and Moscato, P., eds. 2012. *Handbook of Memetic Algorithms*, volume 379 of *Studies in Computational Intelligence*. Springer.

Ong, Y. S.; Lim, M. H.; and Chen, X. S. 2010. Research frontier: Memetic computation - past, present & future. *IEEE Computational Intelligence Magazine* 5(2):24–36.

Paruchuri, P.; Pearce, J. P.; Marecki, J.; Tambe, M.; Ordonez, F.; and Kraus, S. 2008. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *International Conference on Autonomous Agents and Multiagent Systems*, 895–902.

Wang, X.; An, B.; Strobel, M.; and Kong, F. 2018. Catching Captain Jack: Efficient time and space dependent patrols to combat oil-siphoning in international waters. In *AAAI Conference on Artificial Intelligence*, 208–215.

Xu, H.; Ford, B.; Fang, F.; Dilkina, B.; Plumptre, A.; Tambe, M.; Driciru, M.; Wanyama, F.; Rwetsiba, A.; Nsubaga, M.; et al. 2017. Optimal patrol planning for green security games with black-box attackers. In *International Conference on Decision and Game Theory for Security*, 458–477.