

Computing Time-Dependent Policies for Patrolling Games with Mobile Targets

Branislav Bošanský, Viliam Lisý, Michal Jakob, Michal Pěchouček
Agent Technology Center, Dept. of Cybernetics, FEE, Czech Technical University
Technická 2, 16627 Prague 6, Czech Republic
{bosansky, lisy, jakob, pechoucek}@agents.felk.cvut.cz

ABSTRACT

We study how a mobile defender should patrol an area to protect multiple valuable targets from being attacked by an attacker. In contrast to existing approaches, which assume stationary targets, we allow the targets to move through the area according to an a priori known, deterministic movement schedules. We represent the patrol area by a graph of arbitrary topology and do not put any restrictions on the movement schedules. We assume the attacker can observe the defender and has full knowledge of the strategy the defender employs. We construct a game-theoretic formulation and seek defender's optimal randomized strategy in a Stackelberg equilibrium of the game. We formulate the computation of the strategy as a mathematical program whose solution corresponds to an optimal time-dependent Markov policy for the defender. We also consider a simplified formulation allowing only stationary defender's policies which are generally less effective but are computationally significantly cheaper to obtain. We provide experimental evaluation examining this trade-off on a set of test problems covering various topologies of the patrol area and various movement schedules of the targets.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Algorithms, Economics, Experimentation

Keywords

patrolling game, Stackelberg equilibrium, mobile targets, game theory, mathematical programming

1. INTRODUCTION

Game theoretical models have been recently used for modeling scenarios, in which a group of agents (termed *defenders* or *patrollers*) need to protect an area, or prevent an attack on high-value targets. Game theory is a suitable framework for such models as the solutions it provides are optimal

Cite as: Computing Time-Dependent Policies for Patrolling Games with Mobile Targets, , *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. XXX-XXX.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

strategies for the defenders given the opponents' information, capabilities and intentions. Moreover, game-theoretic models have already been successfully applied in real-world security scenarios [11, 10].

Existing approaches address the problem of protecting the targets either by optimizing static allocation of available resources to the targets in order to discover the attacker [8, 7], or by computing the optimal movement strategies for a mobile patroller(s) aiming to interrupt a durative attack on a target in a fully stationary environment [1, 3]. In this paper, we study a problem based on the second category, but – in contrast to the previous work – we assume that the high-value targets can *change their positions* in time.

There are a number of real-world scenarios where the computation of optimal movement strategies for patrolling areas with mobile targets is needed. A typical example from the maritime domain concerns a protection of vessels transiting waters with high pirate activity. Another example concerns unmanned aerial vehicle-based surveillance protecting moving ground targets.

We model the confrontation between the defender (patroller) and the attacker as a two-player non-zero-sum game played on a general directed graph. The movement schedules of the targets are fixed a priori and known to both players. We seek the optimum patrolling strategy as a Strong Stackelberg Equilibrium of the game [12]. This reflects the worst case often present in real-world situations where the attacker is able to observe the defender and its current position, and exploit this information for planning the attack.

Introduction of the target movement requires us to extend the existing work in several important ways. The most fundamental is the ability to use *time-dependent* patrolling policies (i.e. policy changing in time), in contrast to stationary policies (i.e. policy not changing in time), which are only used and sufficient for the case of stationary targets. The introduction of time-dependent policies necessitates the extension of the respective game formulation and, more importantly, novel, formulation of non-linear mathematical programs used for computing solutions of such a game.

We start in the next section by reviewing the previous work on patrolling and security games. In Section 3, we formally define the patrolling game with mobile targets and the solution we seek. The main algorithmic technique we use to solve the game is non-linear optimization; hence, in the sections following, we formulate mathematical programs (MP) that define game solutions for the case with stationary (Section 4) and mobile (Section 5) targets. In Section 6, we

discuss how these highly-complex mathematical programs can be solved using existing solvers and we discuss some solver-independent optimizations. Finally, Section 7 evaluates the quality of the solutions produced and the scalability of our approach on a series of experiments.

2. RELATED WORK

Two main classes of game-theoretic models are dealing with protecting targets or infrastructure from attacks of an adversary: *security games* and *patrolling games*. The main common features of the games are (1) the presence of two players – the defender and the attacker; (2) a very limited amount of resources available for the task – the defender usually cannot guarantee preventing all the attacks, but it optimizes a utility based on the probability of a successful attack; (3) both classes seek the solution mostly in the form of a Stackelberg equilibrium – they seek a strategy that is efficient even if it is known to the attacker.

Security Games. In security games [8] the defender allocates resources to protect the targets according to a randomized strategy. The attacker can observe the strategy of the defender, but cannot observe the current state of the game – i.e. cannot react on the current allocation. The earlier works focused on finding an allocation that minimizes the chance for attacking an unprotected target on large domains [10]. Later works extended the main task with a requirement that the allocation needs to satisfy a set of constraints [11].

Patrolling Games. In the patrolling games the defender moves through an area according to a strategy, while the attacker can observe the current position as well as the strategy of the defender and in the right moment starts attacking some of the targets. The attack takes some time and the goal of the defender is to interrupt this attack.

In [1], the problem of patrolling a perimeter is analyzed. The patrolled environment is modeled as a circle graph, where each node is a potential target. The authors seek the defender’s strategy both as a simple Markovian policy and as a policy with an additional internal state. The implications of limiting the attacker’s knowledge on the same game model are analyzed in [2].

The methods for perimeter patrol cannot be directly applied for patrolling environments with more general topology hence the problem of patrolling on general graphs was studied in a sequence of works by Basilico et al. In [3] the authors define the patrolling problem on an arbitrary graph and provide a general model (termed BGA model) for finding the optimal strategy for the defender. The strategy is defined as a higher-order Markovian policy, though for computational reasons, only experiments with a first-order Markovian policy were performed. Further work in this line of research includes the analysis of the impact of the attacker’s knowledge about the defender’s policy on a general graph [4] and an extension of the model for multiple patrollers [5].

In this paper, we adopt the BGA model and further improve it in order to find optimal strategies for protecting mobile targets. We seek the strategies in the form of the first-order Markovian policy, which has been shown to work well for similar problems [1, 3].

3. PROBLEM DEFINITION

We model the problem of protecting mobile targets as a two-player game between a defender and an attacker.

Environment. The game is played on a directed graph $G = (V, E)$, where the targets Q and the defender can be positioned in any of the vertices. We assume the set E is represented as an adjacency matrix $(e_{i,j})$, where $e_{i,j} = 1$ if there exists an edge from vertex i to j , $\{i, j\} \in E$, and $e_{i,j} = 0$ otherwise. The game is played in turns and we denote the set of turns T , indexed $t = 1 \dots |T|$. In each turn the defender and the targets can move to another vertex. The defender can move only to an adjacent vertex. Contrary, the movement of the targets in the graph can be defined by an arbitrary function $f : Q \times T \mapsto V$. In some scenarios it can be desirable to repeat the game each $|T|$ turns (e.g. targets can move in cycles), hence although the actual number of the turns of the game is higher, we assume that the function f contains operator modulo $|T|$. We refer to this variant as a *repeated version* of the patrolling game. The movement schedule of the targets is a fixed property of the environment and cannot be influenced by any of the players. We further assume that a successful attack on a target takes d turns. The full information about the graph structure (G), the targets’ actual positions and movement schedules (f) is known to both players.

Strategies. The goal of the defender is to move on the graph and to intercept an attack of the attacker, i.e. to come to a node where the attack is taking place. In this paper, we search for a strategy of the defender in the form of first-order Markovian policy. The policy defines for each $i, j \in V$ and $t \in T$ a value $\alpha_{i,j}^t$ representing the probability that the defender present in vertex i in turn t moves to vertex j . We denote the set of all Markovian policies for the defender Θ_d .

The set of possible actions of the attacker $A_a = \{noop, attack(s,t,q)\}$ represents either the action *noop* (i.e. no attack), or starting the attack on a target q when the defender is in vertex s and it is the t -th turn of the game. If the attacker chooses one of the attack actions, it cannot perform any other actions and for next $d \in \mathbb{N}$ turns it can be captured in the vertices $\{f(q, t+1), \dots, f(q, t+d)\}$. We assume that the attacker has a full knowledge of the stochastic strategy executed by the defender. This simulates the worst case attacker observing the defender for a long time before the attack or obtaining a reliable intelligence. The attacker’s strategy is a response function ($AR : \Theta_d \mapsto A_a$), which selects an action for any of the strategies of the defender. We denote the set of all attacker’s strategies Θ_a .

Utilities. Finally, let us define the utility values for both players for each combination of their strategies. In general, there is a limited number of outcomes of the game. The attacker can either be captured, or it can successfully perform an attack on a target $q \in Q$. Following the BGA Model we define $X_0 \in \mathbb{R}$; $X_0 \geq 0$ to be the reward for the defender when it captures the attacker and $X_q \in \mathbb{R}$; $X_q \leq 0$ to be the loss of the defender when the attacker successfully performs an attack on a target $q \in Q$. Similarly, we define the loss and reward $Y_0 \in \mathbb{R}$; $Y_0 \leq 0$, and $Y_q \in \mathbb{R}$; $Y_q \geq 0$ for the attacker being captured and successfully attacking $q \in Q$, respectively.

The strategy of the defender is stochastic; hence the utility value assigned to a combination of two strategies is the expected utility. The values X_0 and X_q (or Y_0 , Y_q respectively) are weighted by the probability of capturing the attacker (π_σ^q) in case that the defender plays ($\sigma \in \Theta_d$) and the attacker plays $AR(\sigma) \in \Theta_a$, which decides to attack the target q :

$$\begin{aligned}
U_d, U_a &: \Theta_d \times \Theta_a \mapsto \mathbb{R} \\
U_d(\sigma, AR(\sigma)) &= X_0 \pi_\sigma^q + X_q (1 - \pi_\sigma^q) \\
U_a(\sigma, AR(\sigma)) &= Y_0 \pi_\sigma^q + Y_q (1 - \pi_\sigma^q) \\
U_a(\sigma, noop) &= U_d(\sigma, noop) = 0
\end{aligned} \tag{1}$$

Solution The defined problem corresponds to Stackelberg (or leader-follower) games and we search for a solution of the game in the form of a Strong Stackelberg Equilibrium (e.g. in [12]). The formal definition of this notion follows.

DEFINITION 3.1. *A pair of strategies $\langle \sigma, AR \rangle$ forms a Strong Stackelberg Equilibrium (SSE) if they satisfy the following:*

1. *The leader (defender) plays a best-response:*
 $U_d(\sigma, AR(\sigma)) \geq U_d(\sigma', AR(\sigma')), \forall \sigma' \in \Theta_d$
2. *The follower (attacker) plays a best-response:*
 $U_a(\sigma, AR(\sigma)) \geq U_a(\sigma, AR'(\sigma)), \forall \sigma \in \Theta_d, AR' \in \Theta_a$
3. *The follower breaks ties optimally for the leader:*
 $U_d(\sigma, AR(\sigma)) \geq U_d(\sigma, AR'(\sigma))$
 $\forall \sigma \text{ and } \forall AR' \in \Theta_a \text{ satisfying 2.}$

SSE is a very suitable equilibrium for the security applications in the real world. First of all, the strategy in SSE is robust against the worst case opponents that have full knowledge of the strategy the defender is executing. Moreover, in many security games, the defender's solution for SSE is also a strategy in the NE of the game [12]. Hence, it is efficient also in the case that the attacker did not observe the defender's strategy and chose its action rationally only based on the definition of the game.

A solution of the game defined in this section can be deterministic, where either the defender can always protect the targets, or the attacker can always perform a successful attack. In this paper we are interested in non-deterministic solutions, where the defender is forced to randomize the movement to maximize the utility based on a chance of capturing the attacker.

4. PATROLLING STATIONARY TARGETS

We have already mentioned in Section 2 that a similar game with stationary targets has been already studied in literature. The approach taken in [3] is to formulate the game as a set of mathematical programs (MPs) and solve it using an existing mathematical optimization software.

In the first part of this section we describe the formulation of the mathematical program presented in [3] and termed *BGA Model*. Later, we present our improvement of the formulation and in the next section we use this improved version of the program as a basis for MPs describing the patrolling game with mobile targets.

4.1 Stationary Game Formulation

The original BGA Model was designed for games with stationary targets, which is a subclass of the game considered in this paper. In order to define the stationary games in our framework, we use several simplifications. Firstly, we assume that the policy is not changing each turn – i.e. $\alpha_{i,j}^1 = \alpha_{i,j}^2 = \dots = \alpha_{i,j}^T$, hence we can omit the upper index t . Secondly, we assume that the function $f(q, t)$ for target $q \in Q$ is a constant (the target is not moving in turns) hence we can directly use index q as the representation of the vertex where the target is placed. Finally, we omit the time index from attacker's actions $attack(s, q)$.

4.2 BGA Model

The BGA Model uses bilinear MPs for computing the policy for the stationary version of our game. Besides the variables for the policy, the programs use helper variables $\gamma_{i,j}^{h,q}$ representing the probability that the defender would reach vertex $j \in V$ beginning in vertex $i \in V$ in exactly $h \in \mathbb{N}$ steps while **not visiting** target $q \in Q$.

As described in [3], the algorithm that uses the BGA Model has two main stages. We omit the mathematical program representing the first stage as it can be easily derived from program in the second stage. In the first stage the algorithm checks whether there exist a defender's strategy, for which the action *wait* would be the best response (i.e. the attacker cannot gain anything by attacking any target). If such a strategy exists, the resulting policy $\sigma = (\alpha_{i,j}; i, j \in V)$ represents the optimal patrolling strategy for the defender. In the other case, the algorithm using the BGA Model enters the second stage where a sequence of bilinear programs is solved.

The goal of the BGA Model is to find a policy that is efficient even against the worst attacker's attack which corresponds to the definition of the Strong Stackelberg Equilibrium (see Definition 3.1). Therefore a mathematical program (MP) is constructed and ran for each attacker's action $attack(s, q)$ as the best response. This reflects the motivation of the SSE – the attacker observes the defender and waits until the defender is located in the most convenient place for the attacker (s), and then starts the attack appropriate target (q). The main results of the program are the value of the game for the defender (i.e., maximized function value) and defender's strategy $\sigma = (\alpha_{i,j}; i, j \in V)$. Finally, as the overall solution of the patrolling problem we select those values of $\alpha_{i,j}$ that were found as the solution of the MP with the highest value of the objective function. The algorithm expressing the use of the MPs as sub-methods for finding a SSE is depicted in Figure 1. The formulation of the mathematical program follows.

$$\max_{\sigma} X_q \sum_{j \in V \setminus q} \gamma_{s,j}^{d,q} + X_0 \left(1 - \sum_{j \in V \setminus q} \gamma_{s,j}^{d,q} \right) \tag{2a}$$

$$\alpha_{i,j} \geq 0 \quad \forall i, j \in V \tag{2b}$$

$$\sum_{j \in V} \alpha_{i,j} = 1 \quad \forall i \in V \tag{2c}$$

$$\alpha_{i,j} \leq e_{i,j} \quad \forall i, j \in V \tag{2d}$$

$$\gamma_{i,j}^{1,g} = \alpha_{i,j} \quad \forall i, j \in V; g \in Q, j \neq g \tag{2e}$$

$$\gamma_{i,j}^{h,g} = \sum_{x \in V \setminus g} \left(\gamma_{i,x}^{h-1,g} \alpha_{x,j} \right) \tag{2f}$$

$$\forall i, j \in V; g \in Q, j \neq g; \forall h \in \{2, \dots, d\}$$

$$\begin{aligned}
Y_q \sum_{j \in V \setminus q} \gamma_{s,j}^{d,q} + Y_0 \left(1 - \sum_{j \in V \setminus q} \gamma_{s,j}^{d,q} \right) &\geq \\
\geq Y_w \sum_{j \in V \setminus g} \gamma_{z,j}^{d,g} + Y_0 \left(1 - \sum_{j \in V \setminus g} \gamma_{z,j}^{d,g} \right) &\tag{2g} \\
&\forall z \in V; g \in Q
\end{aligned}$$

The first two constraints (2b),(2c) ensure that the probabilities $\alpha_{i,j}$ represent a correct defender's policy σ ; (2d) ensure that the defender moves only between two adjacent vertices; constraints (2e)-(2f) recursively define the helper

Input: $G = (V, E)$ – graph; Q – targets
Output: σ – defender’s strategy, v – strategy value
1: **for** $(s, q) \in V \times Q$ **do**
2: $(v, \sigma) = MP(s, q)$
3: **if** $v > v_{max}$ **then**
4: $v_{max} := v$; $\sigma_{max} := \sigma$
5: **end if**
6: **end for**
7: **return** (σ_{max}, v_{max})

Figure 1: The algorithm for computing the defender’s policy for the game.

variables $\gamma_{i,j}^{h,g}$ as the probability of not reaching target g . Finally, constraints (2g) ensure that no other action $attack(z, w)$ gives the attacker a higher expected utility value than the action $attack(s, q)$ for which the program was constructed. Note, that by modifying these constraints in the way that expected utility value of the action $attack(s, q)$ cannot be larger than 0 we obtain the program for the first stage of the algorithm.

The objective function (2a) maximizes the defender’s expected utility U_d . The term $(1 - \sum_{j \in V \setminus q} \gamma_{s,j}^{d,q})$ expresses the probability π_σ^q that the defender (placed in the vertex s) would catch the attacker (attacking the target q).

The BGA Model requires that we construct up to $|V| \times |Q|$ bilinear programs as defined above. The size of the program is quite large as it consists of $O(|V|^3 \cdot d)$ constraints and variables. Moreover, we aim to extend the program to be applicable also for the game with moving targets. Adding the dimension of time to the variables would further increase the size of the program. Therefore we first introduce a reformulation of the BGA Model that lowers the number of variables and constraints in the program.

4.3 Improved BGA Model

Let us now present our novel improvement of the BGA Model, which we later use as the basis for our solution for the problem with mobile targets. All following algorithms have similar two-stage structure as described in Section 4.2. However, for explanatory reasons we further focus only on the second stage and assume that the program solved in the first stage is not feasible. As shown in the previous section the program for the second stage can be easily derived from the presented programs for the second stage.

In order to reduce the number of constraints and variables we remove the variables $\gamma_{i,j}^{h,g}$ from the model and we define an alternative set of variables $\delta_{i,q}^h$, which represent the probability that the defender positioned in node i **reaches**¹ the target q in exactly $h \in \mathbb{N}$ steps. In order to make the formulas even more readable, we further define variables $\omega_{i,q}$ representing the probability that the defender positioned in vertex i visits the target q in *at most* d steps.

Now, we can modify the constraints (2e) - (2g) and optimization function 2a as follows. Again, we formulate one bilinear program for each action $attack(s, q)$ for all $s \in V$, $q \in Q$ being the best response of the attacker. The main results of the program are again the value of the game for

¹Note that the variable δ represent the probability that the defender will visit specific target in comparison to the original probability γ that the defender will not visit the target.

the defender (i.e., maximized function value) and defender’s strategy $\sigma = (\alpha_{i,j}; i, j \in V)$.

$$\max_{\sigma} X_q (1 - \omega_{s,q}) + X_0 \omega_{s,q} \quad (3a)$$

constraints (2b) - (2d)

$$\delta_{i,j}^1 = \alpha_{i,j} \quad \forall i, j \in V \quad (3b)$$

$$\delta_{i,j}^h = \sum_{x \in V \setminus j} (\alpha_{i,x} \delta_{x,j}^{h-1}) \quad \forall i, j \in V; h \in \{2, \dots, d\} \quad (3c)$$

$$\omega_{i,q} = \sum_{h=1}^d \delta_{i,q}^h \quad \forall i \in V; q \in Q \quad (3d)$$

$$Y_q (1 - \omega_{s,q}) + Y_0 \omega_{s,q} \geq Y_g (1 - \omega_{s',q'}) + Y_0 \omega_{s',q'} \quad \forall s' \in V; q' \in Q \quad (3e)$$

The objective function (3a) again maximizes expected defender’s utility function U_d , where the probability of catching the attacker in target q by the defender starting in vertex s is $\pi_\sigma^q = \omega_{s,q}$. The next two constraints (3b)-(3c) define the probability $\delta_{i,j}^h$ using the policy $\sigma = (\alpha_{i,j}; i, j \in V)$. If $h = 1$, then it is exactly the probability connecting the current position of the defender i and the vertex j of the target in the policy σ . For higher h , it is the probability of moving from the current position to some node x (different from the target vertex j) multiplied with the probability of visiting the target vertex j from the node x in exactly $h - 1$ steps. The constraints (3d) defines a helper variable ω , and constraints (3e) again ensure that no other action $attack(z, w)$ gives the attacker a higher expected utility value than the action $attack(s, q)$ for which the program was constructed.

Note, that the reformulation of the probability lowers the size of the program in terms of variables and constraints to $O(|V|^2 \cdot d)$. The solution of the program 3 is the same than in the original program 2. The probability $\omega_{s,q}$ that the defender starting in s *does* visit the target in at most d time steps is the complement of the probability $\gamma_{i,j}^{h,q}$ of not visiting the target q in the original formulation.

5. PATROLLING MOBILE TARGETS

The previous problem formulations assumed that the targets, which the defender tries to periodically visit, statically reside in some vertices of the graph. Further, we assume that these targets change their positions over time based on function $f : Q \times T \mapsto V$ as defined in Section 3. Note that $q \in Q$ cannot be used to identify a node anymore. Further we show that the MP formulation from Section 4.3 can be modified to compute policies even in this dynamic case. There are two main extensions in comparison to the model presented in the previous section: (1) we add the time dimension to the policy and (2) we add the time dimension to the helper variables in the program.

The MP we design in this section searches for an optimal time-dependent policy $\sigma = (\alpha_{i,j}^t; i, j \in V; t \in T)$ for the defender. The reason for using time-dependent policy is that the defender can have substantially different strategy in the same node in different time steps because of the changed positions of the targets. The main helper variable after adding the time dimension has the form $\delta_{s,q}^{h,t}$, with the meaning of the probability that the defender positioned in the vertex $s \in V$ reaches the target $q \in Q$ in exactly $h \in \mathbb{N}$ steps while starting in the t -th ($t \in T$) turn of the game.

As in the previous model, we construct one MP for each attacker’s action and choose the strategy from the MP with

the maximal value for the defender. Compared to the stationary case, the attacker's action $attack(s, t, q)$ depends also on time – i.e. the attacker waits for the “right moment” uniquely identified by the position of the defender $s \in V$ and turn of the game $t \in T$. Then it starts attack on target $q \in Q$. The algorithm of using MPs is similar to the algorithm in Figure 1 and the difference is only in adding the index of the turn of the game.

Each call of the *MP* in the algorithm optimizes the defender's policy $\sigma = (\alpha_{i,j}^t; i, j \in V; t \in T)$ under the assumption that $attack(s, q, t)$ is the optimal action of the attacker. The formulation of the MP for single configuration (s, q, t) is following.

$$\max_{\sigma} X_q (1 - \omega_{s,q}^t) + X_0 \omega_{s,q}^t \quad (4a)$$

$$\alpha_{i,j}^l \geq 0 \quad \forall i, j \in V; l \in T \quad (4b)$$

$$\sum_{j \in V} \alpha_{i,j}^l = 1 \quad \forall i \in V; l \in T \quad (4c)$$

$$\alpha_{i,j}^l \leq e_{i,j} \quad \forall i, j \in V; l \in T \quad (4d)$$

$$\delta_{i,g}^{1;l} = \alpha_{i,f(g,l+1)}^l \quad \forall i \in V; g \in G; l \in T \quad (4e)$$

$$\delta_{i,g}^{h;l} = \sum_{x \in V \setminus f(g,l+1)} \left(\alpha_{i,x}^l \delta_{x,g}^{h-1,((l+1) \bmod |T|)} \right) \quad (4f)$$

$\forall i \in V; g \in Q; h \in \{2, \dots, d\}; l \in T$

$$\omega_{i,g}^l = \sum_{h=1}^d \delta_{i,g}^{h;l} \quad \forall i \in V; g \in Q; l \in T \quad (4g)$$

$$Y_q (1 - \omega_{s,q}^t) + Y_0 \omega_{s,q}^t \geq Y_{q'} (1 - \omega_{s',q'}^{t'}) + Y_0 \omega_{s',q'}^{t'} \quad (4h)$$

$\forall s' \in V; q' \in Q; t' \in T$

The constraints are very similar to improved stationary program 3. Constraints (4b) and (4d) again ensure that σ is a correct policy, and constraints (4e)-(4f) define the probability $\delta_{i,g}^{h;l}$ using the policy σ . The difference is in expressing the vertex of the target using the function f . If $h = 1$, δ is equal to probability connecting the current position of the defender i and the position of the target in the next turn $f(g, l + 1)$. For higher h , it is the probability is calculated similarly to the stationary case, but the excluding vertex is the vertex, where target q is in the next turn $l + 1$. The constraints (4g) define variable ω and constraints (4h) ensure that no alternative attacker strategy can provide higher attacker's utility U_a . The optimized function (4a) is also very similar to the stationary case and it express the expected utility U_d of the patroller's policy σ for a fixed combination of $s \in V, q \in Q$ and $t \in T$.

6. SOLVING THE PROGRAM

If some solver can optimally solve the programs defined above, we would have the optimal strategies for the patrolling problem. However, solving this program is hard. The number of program constraints and variables in the improved stationary formulation is $O(|V|^2 \cdot d)$ and $O(|V|^2 \cdot |T| \cdot d)$ in the time-dependent case. Most of the constraints are bilinear; the remaining constraints as well as the optimized function are linear.

6.1 Alternative Program Formulations

The formulation of the programs in previous sections was chosen with the readability as the main criterion. However, the exact form of the formulation can influence the computational complexity of solving the problem optimally as well

as the potential for approximation. A different formulation of the problem can be constructed if some of the program variables are not represented explicitly in the program.

Bilinear MP The presented form of the programs expresses the optimization of a linear function over a region defined by (at worst) bilinear constraints. The size of the program is polynomial in the relevant problem parameters. However, solving a bilinear program is in general NP-hard [6]. Non-convexity of the feasible region that is defined by the bilinear equalities indicates that this particular problem is most likely not an exception. On the other hand, these programs are widely studied and many approximation algorithms are available.

Polynomial MP Some of the variables in the presented programs do not have to be represented explicitly in an actual program formulation. For example, the variables ω in programs (3) and (4) can be clearly removed and all its occurrences can be substituted by the corresponding sum of variables δ . This modification still leads to a bilinear program. However, if we also remove the variables δ in the same way, we are in a different class of MPs. All the bilinear constraints are removed and only the linear constraints remain. However, the complexity of the optimized function increases dramatically. Instead of linear, it becomes polynomial with maximal degree d . As mentioned in [9], even unconstrained optimization of 4-degree polynomials is NP-hard, hence this formulation is also not likely to produce optimal solution for larger problems in reasonable time.

6.2 Approximate MP Solutions

The discussion above indicates that finding reasonably fast solvers that would solve the presented MPs optimally is unlikely. However, this section shows that even approximation algorithms that do not guarantee finding the optimal result are usable for finding a good solution of the game. In order to do that, we use the most general case of finding the time-dependent policy for mobile targets. The same results hold also for the simpler cases. Let $MP^*(s, q, t)$ be the optimal solution for the program for the setting and $MP(s, q, t)$ be a feasible approximate solution. First of all, we show that any feasible solution of the program provides a strategy with a guaranteed quality.

LEMMA 6.1. *Let $(v, \sigma) = MP(s, q, t)$ be any feasible solution of program (4) for any $(s, q, t) \in V \times Q \times T$. If the attacker plays rationally and the defender uses the strategy σ , it is guaranteed to achieve the utility v .*

PROOF. For any $s, q, t \in V \times Q \times T$ and a feasible strategy σ , the constraints (4g) ensure that the best rational response of the attacker to strategy α is to use the strategy s, q, t . Any other strategy leads to at most the same utility for the attacker. Moreover, according to Definition 3.1, the attacker chooses among its alternative best responses the one that is best for the defender. \square

We continue by showing the relation between the quality of the solution for individual mathematical programs (4) and the quality of the solution produced by Algorithm 1.

LEMMA 6.2. *Let v^* be the value of the optimal strategy of the defender. Assume that each of the programs for different settings of $s, q, t \in V \times Q \times T$ is approximately solved, such that the difference between the defender's utility from the*

produced policy and the optimal policy for the setting is lower than ϵ . Then the difference between the utility of the policy produced by Algorithm 1 and v^* is lower than ϵ .

PROOF. Assume that Algorithm 1 selects the result of $MP(s, q, t)$ to be the output of the whole process. There are two cases we need to consider.

1. $(v^*, \sigma^*) = MP^*(s, q, t)$:
The difference between the produced solution and the optimum is less than ϵ from its definition.
2. $(v^*, \sigma^*) = MP^*(s', q', t')$ and $(s, q, t) \neq (s', q', t')$:
Let $(v, \sigma) = MP(s, q, t)$ and $(v', \sigma') = MP(s', q', t')$. If Algorithm 1 selected σ then $v \geq v'$. $v' \geq v^* - \epsilon$ from the definition of ϵ . Hence $v \geq v^* - \epsilon$, which means that the produced solution is at most ϵ far from the optimum.

□

7. EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the proposed approach. The focus of the paper is on validation of the novel patrolling game model, hence the focus of the experiments is on the quality of the solutions produced by the proposed non-linear program. As discussed in Section 6, solving the program is NP-hard, therefore we also describe several preliminary optimization techniques (more advanced improvements are planned for future work) that help the solver to converge to reasonable solutions in a reasonable time.

7.1 Experiment Settings

We used the following settings for the experiments: (1) we used two types of graphs – *grid* and *grid with holes*; (2) two targets were present in each setting and we used three different movement schedules for the targets; (3) we simplified the values of the targets and assume that all targets have the same value; (4) we compared the quality of produced time-dependent policies to an approximation calculated as a stationary policy.

7.1.1 Graphs

We conducted the evaluation on two types of graphs inspired by a typical application domains: (1) *grid with holes* (see Figure 2(a) for an example) which may e.g. represent a road network, (2) *full grid* (see Figure 2(b) for an example) that corresponds to discretization of open space, such as ocean surface. In both figures, black nodes represent initial positions of targets and dashed arrows show motion patterns for the targets. In all experiments, targets move once per two defender’s moves; this reflects that the defender is faster than targets.

7.1.2 Targets

As adding more targets did not show any interesting changes in the results, we limit the presentation to experiments with two targets only. Three types of target movement with different implications on the distance between the targets were employed: (1) *alternating* where the distance between the targets is decreasing and increasing in time (see Figures 2(a), 2(b)); (2) *equidistant* is defined only for grid graphs and involves simultaneous movement of targets along the top-most and bottom-most edges of the graph from left to right and back. In Figure 2(b), the target at the bottom

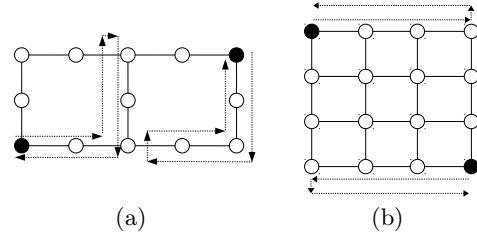


Figure 2: The schema of the experimental scenarios. Black nodes denote target’s initial positions and arrows depict target’s movement.

starts from the left side and moves in the same way as the one on top; (3) *stationary* where targets remain in their initial positions. Finally, in all experiments we adopt the repeated version of the game – i.e. the targets are moving in cycles and the game repeats each $|T|$ turns.

7.1.3 Program for Time-Dependant Policies

For explanatory reasons we simplified the values of the targets, that neither the attacker nor the defender has any preference among the targets – the attacker tries to maximize the probability that it will successfully attack some target and the defender aims to minimize this probability. This corresponds to an instance of the defined patrolling problem where $X_q = -Y_q = -1; \forall q \in Q$ and $Y_0 = X_0 = 0$. As we want to evaluate the probability of catching the attacker we assume that the attacker has to attack some target (i.e. we disallow *noop* action for the attacker).

Using above simplification the game became a zero-sum variant of the original problem, however, it does not substantially change the characteristics of the program, nor it is significantly computationally easier to solve compared to the original formulation due to the non-linearity in constraints of the MPs (δ variables). The only change is the simplification of the objective function and utility-based constraints, which enable us to formulate the MP as a single *min-max* optimization instead of a sequence of optimizations of MPs for each initial point, turn, and target. We are searching for σ that optimizes:

$$\max_{\sigma} \min_{s, q, t} (\omega_{s, q}^t) \quad (5)$$

$$\text{s.t. (4b)-(4g)}$$

Constraints (4h) are substituted by the maximization of the objective function and can be removed. We further refer to the value of the objective function (5) as the *reached value of the game*.

7.1.4 Program for Stationary Policy

In order to evaluate the quality of the solutions based on a time-dependent policy we need to obtain a stationary policy, which still can be efficient even with moving targets in some cases (e.g. if the movement is limited). We compare the performance of these two formulations in terms of the reached game value and computation time in the game with moving targets.

In order to obtain a stationary policy, we have slightly modified the MP (5) – we have removed the time index from all α variables in all constraints. All δ and ω variables keep the time index in order to take target’s movement into account. This modification is especially useful if the variables δ and ω are not explicitly represented in the implementation

Graph	Mov. Type	Policy	d	value	time [s]
grid 4x4	alternating	stationary	8	0.19	6.60
		dynamic	8	0.50	3516.20
		stationary	9	0.33	30.81
		dynamic	9	0.89	14063.46
	equidistant	stationary	9	0.32	37.32
		dynamic	9	0.50	333.22
grid-hole n13	alternating	stationary	8	0.17	4.83
		dynamic	8	0.50	3194.19
		stationary	9	0.26	13.58
		dynamic	9	1.00	9859.08

Table 1: Comparison of the reached value of the game (equals the probability that the defender catches the attacker) and the average computation time; d denotes attack duration.

of the program. In that case, the number of real variables in the program decreases significantly.

Besides the comparison reasons we used the stationary policy as an initial point for solver for calculating the time-dependent policy (see Section 7.2.1).

7.1.5 Implementation

We implemented the proposed mathematical programs in MATLAB[®] using the *fminimax* function for the optimization. For both programs – the MP for the time-dependent and for the stationary policy – we use only α as variables; variables δ and ω are not explicitly represented as variables of the MP. The set of α variables is limited to those $\alpha_{i,j}^t$ for which there exists an edge between vertices.

Internal MATLAB parallel methods were used during the optimization, hence the duration of the experiments is expressed in the total CPU time (in seconds) consumed on all cores.

7.2 Results

In this section we present the results of the experimental evaluation. In general, the results proved that in the game with mobile targets it is reasonable to use the time-dependent policy. In most of the experimental settings usage of time-dependent policy led to significantly higher utility value than the approximation using a stationary policy but currently at the expense of significantly higher computational costs.

The most representative results, in terms of the reached game value and the average computation time, from two graphs (shown in Figures 2(a) and 2(b)) were selected and depicted in Table 1. Note that the value reached in the zero-sum variant represents the worst-case probability that the defender catches the attacker during the attack on some target. As expected, the dynamic policy is significantly better than the stationary approximation, as the defender can better adapt to the movement of the targets. For the third target movement type, i.e. stationary targets, both methods converged to the same values and policies.

The frequent appearance of 0.5 as the reached game value in Table 1 stems from having two targets. In many settings, the defender cannot protect both targets and thus it non-deterministically “chooses” just one of them; the attacker then succeeds if it attacks the other target. Note that the MP also found a deterministic policy that always leads to catching the attacker (reached value is 1.00).

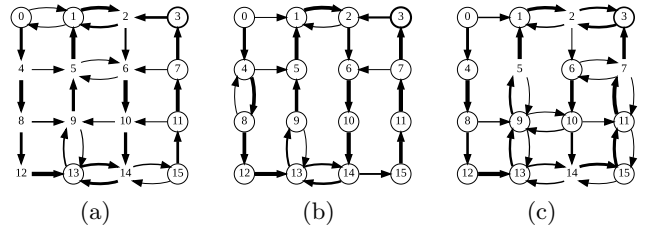


Figure 3: Defender’s policies. Two targets move right from vertices (0,12) to (3,15) and back. The probability of using an edge corresponds to thickness of the respective edge or circle (in the case of loops). A stationary policy (Figure (a)) and two snapshots of a time-dependent policy are shown – turn 6 with targets at (2, 14) (Figure (b)) and turn 7 with targets at (3, 15) (Figure (c)).

The differences between stationary policy and time-dependent policies for the defender can be seen in Figure 3: the stationary policy 3(a) covers all positions of the targets in time, while the time-dependent policy can utilize the knowledge of the current positions of targets (vertices 2 and 14 in 3(b) showing turn 6) and also future positions of targets (3(c) shows turn 7 of the game with targets in vertices 3 and 15). Note, that thanks to the time-dependant policy, the defender can in turn 7 reach the target in vertex 3 from the vertex 2 in one move, however, there is no such possibility in the stationary policy.

7.2.1 Initial Values for Computing Time-Dependent Policies

In Section 7.1.4 we mentioned that the approximate solution of the problem using a stationary policy can be used by the solver as the initial point for searching for a time-dependant policy. In Figure 4, we compare the computation time and the reached value of the game for a fixed graph (grid 3x4, with $d = 6$) with different initial points. Note that the graph is in logarithmic scale. When a random policy is used for initialization (circle), most runs of the solver were very quick but unsuccessful (i.e. the optimization stopped in a local minimum with low value). For random values not representing a legal policy (cross), most of runs stopped in a local minimum with low value as well, but they took significantly more time. Finally, when using the stationary policy approximation as the initial value (diamond), the runtime is comparable to random non-policy initialization and the reached game value is maximal. Pattern visible in Figure 4 was observed for other graphs as well.

7.2.2 Scalability

To address the scalability of the approach, we performed experiments (see Table 2) on grid graphs with an expanding proportion. We can see that average time to solve the program is increasing exponentially for both stationary and dynamic policy. However, we had not implemented any significant improvements leading to simplifying the mathematical programs and (s, q, t) configurations, hence there is a possibility for significant improvement of performance of proposed approach.

8. CONCLUSION

We presented a novel formal model – a patrolling game with mobile targets. It is a two-player game between the

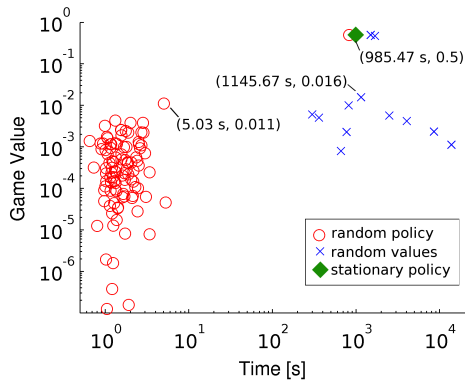


Figure 4: Consumed time (x-axis) and reached value (y-axis) for different initialization of the solver computing the time-dependant policy. Both axes use logarithmic scale.

Graph	Policy	d	value	time [s]
grid 2x4	stationary	4	0.12	1.06
	dynamic		0.50	122.44
grid 3x4	stationary	6	0.18	17.97
	dynamic		0.50	985.47
grid 4x4	stationary	8	0.24	29.77
	dynamic		0.50	3516.20
grid 5x4	stationary	10	0.27	55.08
	dynamic		0.66	53057.10

Table 2: Results of scale-up experiments (increasing the height of the grid).

defender, patrolling in an area in order to protect a set of targets, and the attacker who wants to attack the targets. We assume that the attacker has full knowledge about the defender’s strategy, and that an attack takes non-zero time to complete, during which the attacker can be discovered by the defender. In contrast to the existing work in the domain of patrolling games, we allow the targets to move through the area.

We provided a formal definition of this novel patrolling game and a mathematical program for finding defender’s optimal strategy, sought as the game’s Strong Stackelberg Equilibrium. Specifically, we search for a time-dependent Markovian policy for the defender that utilizes the knowledge of the movement schedule of the targets. As the mathematical program is non-linear, finding the solution is computationally hard. We therefore performed several experiments to evaluate the proposed approach. The results justify using time-dependent policies in scenarios with moving targets, as the reached value of the game, i.e. the utility of the defender, was significantly higher compared to the situations, where defender’s strategies are limited to stationary policies.

Our results open a number of future work directions. Currently, we provided only a basic implementation of the proposed program and further improvements are desirable to improve the scalability of the approach. Furthermore, the investigation of time-dependent non-markovian policies (i.e. where the defender has some internal state e.g. representing the target, towards which the defender is heading) can further enrich the space of patrolling games. Moreover, the observability of the defender’s internal state by the attacker can reflect the imperfectness of the attacker’s knowledge.

Finally, a more compact representation of the environment of the game (e.g. only in terms of relative distances to the targets) might be employed to reduce the complexity of computation, in particular when combined with non-markovian policies of the defender.

9. ACKNOWLEDGEMENTS

The work presented is supported by the U.S. Army grant no. W15P7T-05-R-P209 (contract BAA 8020902.A), by the Office of Naval Research project no. N00014-09-1-0537, by the Czech Ministry of Education, Youth and Sports under Research Programme no. MSM6840770038: Decision Making and Control for Manufacturing III, and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS10/188/OHK3/2T/13.

10. REFERENCES

- [1] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, 2008.
- [2] N. Agmon, V. Sadv, G. A. Kaminka, and S. Kraus. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *AAMAS*, pages 55–62, 2008.
- [3] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, pages 57–64, 2009.
- [4] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *WI-IAT*, pages 557–564, 2009.
- [5] N. Basilico, N. Gatti, and F. Villa. Asynchronous Multi-Robot Patrolling against Intrusion in Arbitrary Topologies. In *AAAI*, 2010.
- [6] K. Bennett and O. Mangasarian. Bilinear separation of two sets inn-space. *Computational Optimization and Applications*, 2(3):207–227, 1993.
- [7] M. Jain, E. Karde, C. Kiekintveld, F. Ordóñez, and M. Tambe. Optimal defender allocation for massive security games: A branch and price approach. In *Workshop on Optimization in Multi-Agent Systems at AAMAS*, 2010.
- [8] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, pages 689–696, 2009.
- [9] J. Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on Optimization*, 11(3):796–817, 2001.
- [10] J. Pita, M. Jain, J. Marecki, F. Ordó nez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR protection: the application of a game theoretic model for security at the Los Angeles Int. Airport. In *AAMAS*, pages 125–132, 2008.
- [11] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - A Tool for Strategic Security Allocation in Transportation Networks Categories and Subject Descriptors. In *AAMAS*, pages 37–44, 2009.
- [12] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, pages 1139–1146, 2010.