A Security Game Model for Environment Protection in the Presence of an Alarm System

Nicola Basilico[1(✉)], Giuseppe De Nittis[2], and Nicola Gatti[2]

[1] Department of Computer Science, University of Milan, Milan, Italy
nicola.basilico@unimi.it
[2] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milan, Italy

Abstract. We propose, to the best of our knowledge, the first Security Game where a *Defender* is supported by a spatially uncertain *alarm system* which non–deterministically generates *signals* once a target is under attack. Spatial uncertainty is common when securing large environments, e.g., for wildlife protection. We show that finding the equilibrium for this game is $\mathcal{FNP}$–hard even in the zero–sum case and we provide both an exact algorithm and a heuristic algorithm to deal with it. Without false positives and missed detections, the best patrolling strategy reduces to stay in a place, wait for a signal, and respond to it at best. This strategy is optimal even with non–negligible missed detection rates.

1 Introduction

Security Games model the task of protecting physical environments as a non–cooperative game between a *Defender* and an *Attacker* [12]. Usually taking place under a Stackelberg (a.k.a. leader–follower) paradigm [18], they have been shown to outperform other approaches such as, e.g., MDPs [16] and they have been employed in a number of on–the–field systems [9,13]. Recent research lines aim at refining the models by incorporating features from real–world applications, e.g., in [1,20] the Attacker may have different observation models and limited planning capabilities, in [6] realistic aspects of infrastructures to be protected are taken into account. Patrolling is one of the recently studied applications where the Defender controls mobile resources (such as patrolling robots) and the Attacker aims at compromising some locations denoted as targets [2]. Equilibrium strategies prescribe how the Defender should schedule resources in time to maximize its expected utility.

Infrastructures and environments that need to be surveilled are usually characterized by the presence of locally installed sensory systems. *Detection sensors* are able to gather measurements about suspicious events that an *alarm system* can process to generate alarm *signals*. These physical devices often present some degree of inaccuracy, such as false positives rates or missed detections rates. Alarm signals are spatially uncertain, meaning that they do not precisely localize the detected event, but provide a probabilistic belief over the locations

MHR Khouzani et al. (Eds.): GameSec 2015, LNCS 9406, pp. 192–207, 2015.
DOI: 10.1007/978-3-319-25594-1_11

potentially under attack. Spatial uncertainty is common when dealing with complex infrastructures or large open environments, where a *broad area surveillance* activity, in which an attack is detected but only approximately localized, triggers a *local investigation* activity, where guards have to find and clear the attack. A similar approach is adopted in a number of real–world problems where cheap and spatially uncertain sensors cover the targets to be protected. In [10], the problem of poaching of endangered species is studied and a device to help rangers against this threat is proposed. The introduction of cheap wide–range sensors, affordable by the conservation agencies, could significantly improve the behavior of the rangers, giving them information about the areas in which a potential attack is occurring. Other applications include UAVs surveillance [4], wildfires detection with CCD cameras [14] and monitoring agricultural fields [11]. In [21] a system for surveillance based on wireless sensor networks is proposed.

To the best of our knowledge, [8] is the only work integrating sensors in Security Games. It assumes sensors with no spatial uncertainty in detecting attacks on single targets. When no false positives are possible, an easy variation of the algorithm for the case without sensors [2] can be used, while, when false positives are present, the problem is computationally intractable.

Contributions. In this paper, we propose the first Security Game model that integrates a spatially uncertain alarm system in game–theoretic settings for patrolling. Each alarm signal carries the information about the set of targets that can be under attack and is described by a probability of being generated when each target is attacked. Moreover, the Defender can control only one patroller. We show that finding the equilibrium is $\mathcal{FNP}$–hard even in the zero–sum case and we give an exact exponential–time algorithm and a heuristic algorithm to deal with it. When no false positives and no missed detections are present, the optimal Defender strategy is to stay in a fixed location, wait for a signal, and respond to it at best. This strategy keeps being optimal even when non–negligible missed detection rates are allowed. Finally, we experimentally evaluate the scalability of our exact algorithm and we compare it with respect to the heuristic one in terms of solution quality.

2 Problem Formulation

Basic patrolling security game models [2,19] are turn–based extensive–form games with infinite horizon and imperfect information between two agents: an *Attacker* $\mathcal{A}$ and a *Defender* $\mathcal{D}$. The environment to be patrolled is formally described by an undirected connected graph $G = (V, E)$. Each edge $(i, j) \in E$ requires one turn to be traversed, while we denote with $\omega^*_{i,j}$ the temporal cost (in turns) of the shortest path between any i and $j \in V$. We denote by $T \subseteq V$ the subset of vertices called *targets* that have some value for $\mathcal{D}$ and $\mathcal{A}$. Each target $t \in T$ is characterized by a value $\pi(t) \in (0, 1]$ and a penetration time $d(t) \in \mathbb{N}$ measuring the number of turns needed to complete an attack over t. At each turn of the game, agents $\mathcal{A}$ and $\mathcal{D}$ play simultaneously: if $\mathcal{A}$ has not attacked in the previous turns, it can observe the position of $\mathcal{D}$ in the graph and decides

whether to attack a target[1] or to wait for a turn, while $\mathcal{D}$ has no information about the actions undertaken by $\mathcal{A}$ in previous turns and decides the next vertex to patrol among all those adjacent to the current one. If $\mathcal{D}$ patrols a target t that is under attack of $\mathcal{A}$ before $d(t)$, $\mathcal{A}$ is captured. The game is constant sum (then equivalent to a zero sum game): if $\mathcal{A}$ is captured, $\mathcal{D}$ receives a utility of 1 and $\mathcal{A}$ receives 0, while, if an attack over t has success, $\mathcal{D}$ receives $1 - \pi(t)$ and $\mathcal{A}$ receives $\pi(t)$; finally, if $\mathcal{A}$ waits forever, $\mathcal{D}$ receives 1 and $\mathcal{A}$ receives 0. The appropriate solution concept is the leader–follower equilibrium. The game being constant sum, the best leader's strategy is its maxmin/minmax strategy.

Our *Patrolling Game* (PG) extends the above model introducing a *spatial uncertain alarm system* available to $\mathcal{D}$. The system is defined as a tuple (S, p) where $S = \{s_1, \cdots, s_m\}$ is a set of $m \geq 1$ *signals* and $p : S \times T \rightarrow [0, 1]$ is a function that specifies the probability of having the system generating a signal s given that target t has been attacked. With a slight abuse of notation, for a signal s we define $T(s) = \{t \in T \mid p(s \mid t) > 0\}$ and, similarly, for a target t we have $S(t) = \{s \in S \mid p(s \mid t) > 0\}$. In this work, we initially assume that the alarm system is not affected by false positives, i.e. a signal is generated but no attack has occurred, or missed detections, i.e. the signal is not generated even though an attack has occurred. In our model, at each turn, before deciding its next move, agent $\mathcal{D}$ can observe whether or not a signal has been generated by the alarm system.

We observe that, since no false positive and no missed detection are present, $\mathcal{D}$ will always receive a signal as soon as $\mathcal{A}$ starts an attack. This allows us to identify, in our game model, a number of subgames, each in which $\mathcal{D}$ is in a given vertex v and an attack is started. The solution to our PG can be safely found by, at first, finding the best strategies of $\mathcal{D}$ in responding to a signal from any $v \in V$ and, subsequently, on the basis of such signal–response strategies, by finding the best patrolling strategy over G. In Sect. 3 , we present algorithms to find the best signal–response strategies, while, in Sect. 4, we focus on the best patrolling strategies.

3 Finding the Best Signal–Response Strategy

We study the subgame in which $\mathcal{D}$ is in a vertex v and $\mathcal{A}$ decides to attack. We call it *Signal–Response Game given* v (SRG–v). The actions available to $\mathcal{A}$ are given by T and its strategy $\sigma_v^{\mathcal{A}}$ is defined as a probability distribution over T. We denote with $\sigma_{v,s}^{\mathcal{D}}$ the generic strategy of $\mathcal{D}$ when it is at v and receives a signal s and we discuss below the problem of defining the space of actions available to $\mathcal{D}$. We denote with g_v the expected utility of $\mathcal{A}$, the expected utility of $\mathcal{D}$ is $1 - g_v$. We show that, independently of how we define the space of actions of $\mathcal{D}$, the problem of finding the best $\sigma_v^{\mathcal{D}} = (\sigma_{v,s_1}^{\mathcal{D}}, \ldots, \sigma_{v,s_m}^{\mathcal{D}})$ is $\mathcal{FNP}$–hard [5]. We do this by assessing the complexity of its decision version.

[1] As is customary, we assume that $\mathcal{A}$ can instantly reach the target of its attack. This assumption can be easily relaxed as shown in [3].

Definition 1. *k–***SRG***–v*
INSTANCE: an instance of SRG–v as defined above;
QUESTION: is there $\sigma^{\mathcal{D}}$ *such that* $g_v \leq k$*?*

Theorem 1. *k–***SRG***–v is* $\mathcal{NP}$*–hard.*

Proof. Let us consider the following reduction from HAMILTONIAN–PATH. Given an instance of HAMILTONIAN–PATH $G_H = (V_H, E_H)$, we build an instance for k–**SRG**–v as:

- $V = V_H \cup \{v\}$;
- $E = E_H \cup \{(v, h), \forall h \in V_H\}$;
- $T = V_H$;
- $d(t) = |V_H|$;
- $\pi(t) = 1$, for all $t \in T$;
- $S = \{s\}$;
- $p(s \mid t) = 1$, for all $t \in T$;
- $k = 0$.

If $g_s \leq 0$, then there must exist a path starting from v and visiting all the targets in T by $d = |V_H|$. Given the edge costs and penetration times assigned in the above construction, the path must visit each target exactly once. Therefore, since $T = V_H$, the game's value is less or equal than zero if and only if G_H admits an Hamiltonian path. This concludes the proof. □

Given that an SRG–v is a subgame of the PG, it follows that finding the best strategy of $\mathcal{D}$ in PG is $\mathcal{FNP}$–hard. Since computing maxmin/minmax strategies can be done in polynomial time in the size of the payoffs matrix by means of linear programming, the difficulty of SRG–v resides in the generation of the payoffs matrix whose size is in the worst case exponential in the size of the graph (unless $\mathcal{P} = \mathcal{NP}$).

Now we focus on the problem of defining the set of actions available to $\mathcal{D}$ when it is in v and receives signal s. We define a generic *route* r as a sequence of vertices visited by $\mathcal{D}$. We denote with $r(i)$ the i–th vertex visited along r and with $A_r(r(i)) = \sum_{h=0}^{i-1} \omega^*_{r(h),r(h+1)}$ the time needed by $\mathcal{D}$ to visit $r(i)$ starting from $r(0)$. We restrict our attention on a subset of routes, that we call *covering routes*, with the following properties: $r(0) = v$ (i.e., the starting vertex is v), $\forall i \geq 1$ it holds $r(i) \in T(s)$, where s is the signal generated by the alarm system (i.e., only targets potentially under attack are visited) and $\forall i \geq 1$ it holds $A_r(r(i)) \leq d(r(i))$ (i.e., all the targets are visited within their penetration times) with $\mathcal{D}$ moving on the shortest paths between each pair of targets. Notice that a covering route r may visit a strict subset of $T(s)$. The set of actions available to $\mathcal{D}$ is given by all the covering routes. Given a covering route r, with a slight abuse of notation, we define the *covering set* $T(r)$ as the set of targets visited along r and we denote with $c(r)$ the temporal cost of the corresponding path, that is $c(r) = A_r(r(|T(r)|))$. Notice that in the worst case the number of covering routes is $O(|T(s)|^{|T(s)|})$, but using all of them may be unnecessary since some covering

routes will never be played by $\mathcal{D}$ due to strategy domination and therefore they can be safely discarded [15]. We introduce two definitions of dominance that we use below.

Definition 2 (Intra–Set Dominance). Given two different covering routes r, r' for (v, s) such that $T(r) = T(r')$, if $c(r) \leq c(r')$ then r dominates r'.

Definition 3 (Inter–Set Dominance). Given two different covering routes r, r' for (v, s), if $T(r) \supset T(r')$ then r dominates r'.

Definition 2 suggests that we can safely use only one route per covering set. Covering sets suffice for computing the payoffs matrix of the game and in the worst case are $O(2^{|T(s)|})$, with a remarkable reduction of the search space w.r.t. $O(|T(s)|^{|T(s)|})$. However, any algorithm working directly with covering sets instead of covering routes should also decide whether or not a set of targets is a covering one: this problem is hard.

Definition 4. COV–SET
INSTANCE: a graph $G = (V, E)$, a target set T with penetration times d, and a starting vertex v;
QUESTION: is T a covering set for some covering route r?

By trivially adapting the same reduction for Theorem 1 we can state the following theorem.

Theorem 2. *COV–SET is $\mathcal{NP}$–complete.*

Computing a covering route for a given set of targets (or deciding that no covering route exists) is not doable in polynomial time unless $\mathcal{P} \neq \mathcal{NP}$. In addition, Theorem 2 suggests that no algorithm for COV–SET can have complexity better than $O(2^{|T(s)|})$ unless there is a better algorithm for HAMILTONIAN–PATH than the best algorithm known in the literature. This seems to suggest that enumerating all the possible subsets of targets and, for each of them, checking whether or not it is covering requires a complexity worse than $O(2^{|T(s)|})$. Surprisingly, we show in the next section that there is an algorithm with complexity $O(2^{|T(s)|})$ (neglecting polynomial terms) to enumerate all and only the covering sets and, for each of them, one covering route. Therefore, the complexity of our algorithm matches (neglecting polynomial terms) the complexity of the best known algorithm for HAMILTONIAN–PATH.

Definition 3 suggests that we can reduce further the set of actions available to $\mathcal{D}$. Given a covering set Q (where $Q = T(r)$ for some r), we say that Q is *maximal* if there is no route r' such that $Q \subset T(r')$. In the best case, when there is a route covering all the targets, the number of maximal covering sets is 1, while the number of covering sets is $2^{|T(s)|}$, thus considering only maximal covering sets allows an exponential reduction of the payoffs matrix. In the worst case, when all the possible subsets of $|T(s)|/2$ targets are maximal covering sets, the number of maximal covering sets is $O(2^{|T(s)|-2})$, while the number of covering sets is $O(2^{|T(s)|-1})$, allowing a reduction of the payoffs matrix by a

factor of 2. Furthermore, if we knew *a priori* that Q is a maximal covering set we could avoid to search for covering routes for any set of targets that strictly contains Q. When designing an algorithm to solve this problem, Definition 3 could then be exploited to introduce some kind of pruning technique for saving average compute time. However, the following result shows that deciding whether a covering set is maximal is hard.

Definition 5. MAX–COV–SET
INSTANCE: a graph $G = (V, E)$, a target set (T, d), a starting vertex v, and a covering set $T' \subset T$;
QUESTION: is T' maximal?

Theorem 3. *MAX–COV–SET is in co–$\mathcal{NP}$ and no polynomial time for it exists unless $\mathcal{P} = \mathcal{NP}$.*

Proof. Any covering route r such that $T(r) \supset T'$ is a NO certificate for MAX–COV–SET, placing it in co–$\mathcal{NP}$. (Notice that, due to Theorem 2, having a covering set would not suffice given that we cannot verify in polynomial time whether it is actually covering unless $\mathcal{P} = \mathcal{NP}$.)

Let us suppose we have a polynomial–time algorithm for MAX–COV–SET, called A. Then (since $\mathcal{P} \subseteq \mathcal{NP} \cap \text{co-}\mathcal{NP}$) we have a polynomial algorithm for the complement problem, i.e., deciding whether all the covering routes for T' are dominated. Let us consider the following algorithm: given an instance for COV–SET specified by graph $G = (V, E)$, a set of target T with penetration times d, and a starting vertex v:

1. assign to targets in T a lexicographic order $t_1, t_2, \ldots, t_{|T|}$;
2. for every $t \in T$, verify if $\{t\}$ is a covering set in $O(n)$ time by comparing $\omega^*_{v,t}$ and $d(t)$; if at least one is not a covering set, then output NO and terminate; otherwise set $\hat{T} = \{t_1\}$ and $k = 2$;
3. apply algorithm A on the following instance: graph $G = (V, E)$, target set $\{\hat{T} \cup \{t_k\}, \hat{d}\}$ (where $\hat{d}$ is d restricted to $\hat{T} \cup \{t_k\}$), start vertex v, and covering set $\hat{T}$;
4. if A's output is YES (that is, $\hat{T}$ is not maximal) then set $\hat{T} = \hat{T} \cup \{t_k\}$, $k = k + 1$ and restart from step 3; if A's output is NO and $k = |T|$ then output YES; if A's output is NO and $k < |T|$ then output NO;

Thus, the existence of A would imply the existence of a polynomial algorithm for COV–SET which (under $\mathcal{P} \neq \mathcal{NP}$) would contradict Theorem 2. This concludes the proof. □

Nevertheless, we show in the following section that there is an algorithm enumerating all and only the maximal covering sets and one route for each of them (which potentially leads to an exponential reduction of the time needed for solving the linear program) with only an additional polynomial cost w.r.t. the enumeration of all the covering sets and therefore, neglecting polynomial terms, has a complexity $O(2^{|T(s)|})$.

3.1 Computing $\mathcal{D}$'s actions

Here we provide an algorithm to find the set of actions available to $\mathcal{D}$ when it is in v and receives signal s. Let us denote $C^k_{v,t}$ a collection of covering sets $Q^k_{v,t}$s such that $Q^k_{v,t}$ has cardinality k and admits a covering route r whose starting vertex is v and whose last vertex is t. Each $Q^k_{v,t}$ is associated with a cost $c(Q^k_{v,t})$ representing the temporal cost of the shortest covering route for $Q^k_{v,t}$ that specifies t as the k–th target to visit. Upon this basic structure, our algorithm iteratively computes covering sets collections and costs for increasing cardinalities, that is from $k = 1$ possibly up to $k = |T|$ including one target at each iteration. Using a dynamic programming approach, we assume to have solved up to cardinality $k - 1$ and we specify how to complete the task for cardinality k. Detailed steps are reported in Algorithm 1, while in the following we provide an intuitive description. Given $Q^{k-1}_{v,t}$, we can compute a set of targets Q^+ (Line 6) such that for each target $t' \in Q^+$, $t' \notin Q^{k-1}_{v,t}$ and, if t' is appended to the shortest covering route for $Q^{k-1}_{v,t}$, it will be visited before $d(t')$. If Q^+ is not empty, for each $t' \in Q^+$, we extend $Q^{k-1}_{v,t}$ (Line 8) by including it and naming the resulting covering set as $Q^k_{v,t'}$ since it has cardinality k and we know it admits a covering route with last vertex t'. Such route is obtainable by appending t' to the covering route for $Q^{k-1}_{v,t}$ and has cost $c(Q^{k-1}_{v,t}) + \omega^*_{t,t'}$. This value is assumed to be the cost of the extended covering set. (In Line 9 we make use of a procedure $Search(Q, C)$ which outputs Q if $Q \in C$ and $\emptyset$ otherwise). If such extended covering set is not present in collection $C^k_{v,t'}$ or is already present with a higher cost (Line 10), then collection and cost are updated (Lines 11 and 12). After the iteration for cardinality k is completed, for each covering set Q in collection $C^k_{v,t}$, $c(Q)$ represents the temporal cost of the shortest covering route with t as last target.

Algorithm 1. ComputeCovSets_Basic(v, s)

1: $\forall t \in T(s), k \in \{2, \ldots, |T(s)|\}$, $C^1_{v,t} = \{t\}$, $C^k_{v,t} = \emptyset$
2: $\forall t \in T(s)$, $c(\{t\}) = \omega^*_{v,t}$, $c(\emptyset) = \infty$
3: **for all** $k \in \{2 \ldots |T(s)|\}$ **do**
4: **for all** $t \in T(s)$ **do**
5: **for all** $Q^{k-1}_{v,t} \in C^{k-1}_{v,t}$ **do**
6: $Q^+ = \{t' \in T(s) \setminus Q^{k-1}_{v,t} \mid c(Q^{k-1}_{v,t}) + \omega^*_{t,t'} \leq d(t')\}$
7: **for all** $t' \in Q^+$ **do**
8: $Q^k_{v,t'} = Q^{k-1}_{v,t} \cup \{t'\}$
9: $U = Search(Q^k_{v,t'}, C^k_{v,t'})$
10: **if** $c(U) > c(Q^{k-1}_{v,t}) + \omega^*_{t,t'}$ **then**
11: $C^k_{v,t'} = C^k_{v,t'} \cup \{Q^k_{v,t'}\}$
12: $c(Q^k_{v,t'}) = c(Q^{k-1}_{v,t}) + \omega^*_{t,t'}$
13: **end if**
14: **end for**
15: **end for**
16: **end for**
17: **end for**

After Algorithm 1 completed its execution, for any arbitrary $T' \subseteq T$ we can easily obtain the temporal cost of its shortest covering route as

$$c^*(T') = \min_{Q \in Y_{|T'|}} c(Q)$$

where $Y_{|T'|} = \cup_{t \in T}\{Search(T', C_{v,t}^{|T'|})\}$ (notice that if T' is not a covering set then $c^*(T') = \infty$). Algorithm 1 is dubbed "basic" because it does not specify how to carry out two sub–tasks we describe in the following.

The first one is the *annotation of dominated covering sets*. Each time Lines 11 and 12 are executed, a covering set is added to some collection. Let us call it Q and assume it has cardinality k. Each time a new Q has to be included at cardinality k, we mark all the covering sets at cardinality $k-1$ that are dominated by Q (as per Definition 3). The sets that can be dominated are in the worst case $|Q|$, each of them has to be searched in collection $C_{v,t}^{k-1}$ for each feasible terminal t and, if found, marked as dominated. The number of terminal targets and the cardinality of Q are at most n and the *Search* procedure can be efficiently executed in $O(|T(s)|)$ using a binary tree approach. Therefore, dominated covering sets can be annotated with a $O(|T(s)|^3)$ extra cost at each iteration of Algorithm 1. We can only mark and not delete dominated covering sets since they can generate non–dominated ones.

The second task is the *generation of routes*. Algorithm 1 focuses on covering sets and does not maintain a list of corresponding routes. In fact, to build the payoffs matrix for SRG–v we do not strictly need covering routes since covering sets would suffice to determine payoffs. However, we do need them operatively since $\mathcal{D}$ should know in which order targets have to be covered to physically play an action. This task can be accomplished by maintaining an additional list of routes where each route is obtained by appending terminal vertex t' to the route stored for $Q_{v,t}^{k-1}$ when set $Q_{v,t}^{k-1} \cup \{t'\}$ is included in its corresponding collection. At the end of the algorithm only routes that correspond to non–dominated covering sets are filtered out. Maintaining such a list introduces a $O(1)$ cost.

Algorithm 1, in the worst case, has to compute covering sets up to cardinality $|T(s)|$. The number of operations is then bounded by $\sum_{i=1}^{|T(s)|} \binom{|T(s)|}{i-1} i(|T(s)|-1)$ which is $O(|T(s)|^2 2^{|T(s)|})$. With annotations of dominances and routes generation the whole algorithm yields a worst case complexity of $O(|T(s)|^5 2^{|T(s)|})$.

3.2 A Heuristic Algorithm

We know that no polynomial–time algorithm solves exactly the COV–SET problem (unless $\mathcal{NP} = \mathcal{P}$) and therefore any exact algorithm of our problem cannot scale to tackle large settings. In this section, we focus on the design of a heuristic algorithm that can be used for very large instances of patrolling games with spatially uncertain alarms. We note that even if we had a polynomial–time approximation algorithm for COV–SET we would need to call the algorithm $O(2^{|T(s)|})$ times, one per set of targets, and therefore we would not have a polynomial–time approximation algorithm for our problem. This is why we do not focus on the design of approximation algorithms for COV–SET.

Our heuristic algorithm works as follows. Given v and s, for each target $t \in T(s)$ such that $w^*_{v,t} \leq d(t)$ we generate a covering route r with $r(0) = v$ and $r(1) = t$. Thus, $\mathcal{D}$ has at least one covering route per target (that can be covered in time from v). Each route r is expanded by inserting a target $t' \notin T(s) \setminus T(r)$ after position p and shifting each target that was at position $i > p$ in r at position $i + 1$. The pair (t', p) that determines the next expansion is chosen as the pair maximizing a heuristic function $h_r(t', p)$ among all the pairs leading to covering routes (i.e., insertions that make $A_r(t'') > d(t'')$ for some t'' are excluded). Route r is repeatedly expanded in greedy fashion until no insertion is possible. As a result, our algorithm generates at most $|T(s)|$ covering routes.

The heuristic function is defined as $h_r : \{T(s) \setminus T(r)\} \times \{1 \dots |T(r)|\} \rightarrow \mathbb{Z}$, where $h_r(t', p)$ evaluates the cost of expanding r by inserting target t' after the p–th position of r. The basic idea (inspired by [17]) is to adopt a conservative approach, trying to preserve feasibility. Given a route r, let us define the *possible forward shift* of r as the minimum temporal margin in r between the arrival at a target t and $d(t)$: $PFS(r) = \min_{t \in T(r)}(d(t) - A_r(t))$. The *extra mileage* $e_r(t', p)$ for inserting target t' after position p is the additional traveling cost to be paid: $e_r(t', p) = (A_r(r(t')) + \omega^*_{r(p),t'} + \omega^*_{t',r(p+1)}) - A_r(r(p+1))$. The *advance time* that such insertion gets with respect to $d(t')$ is defined as: $a_r(t', p) = d(t') - (A_r(r(p)) + \omega^*_{r(p),t'})$. Finally, $h_r(t', p)$ is defined as: $h_r(t', p) = \min\{a_r(t', p); (PFS(r) - e_r(t', p))\}$.

We partition the set $T(s)$ in two sets T_{tight} and T_{large} where $t \in T_{tight}$ if $d(t) < \delta \cdot \omega^*_{v,t}$ and $t \in T_{large}$ otherwise ($\delta \in \mathbb{R}$ is a parameter). The previous inequality is a non–binding choice we made to discriminate targets with a tight penetration time from those with a large one. Initially, we insert all the tight targets and only subsequently we insert the non–tight targets. It can be easily observed that our heuristic algorithm runs in $O(|T(s)|^3)$ given that heuristic h_r can be computed in $O(|T(s)|^2)$.

3.3 Solving SRG–v

Now we can formulate the problem of computing the equilibrium signal response strategy for $\mathcal{D}$. Let us denote with $\sigma^{\mathcal{D}}_{v,s}(r)$ the probability with which $\mathcal{D}$ plays route r under signal s and with $R_{v,s}$ the set of all the routes available to $\mathcal{D}$ generated by some algorithm. We introduce function $\mathcal{U}_{\mathcal{A}}(r, t)$ returning $\pi(t)$ if r is not a route covering t and 0 otherwise. The best $\mathcal{D}$ strategy (maxmin strategy) can be found by solving the following linear mathematical programming problem:

$$
\begin{aligned}
& & \min \quad & g_v & \text{s.t.} \\
\sum_{s \in S(t)} p(s \mid t) \sum_{r \in R_{v,s}} & \sigma^{\mathcal{D}}_{v,s}(r) U_{\mathcal{A}}(r, t) \leq g_v & & \forall t \in T \\
& \sum_{r \in R_{v,s}} \sigma^{\mathcal{D}}_{v,s}(r) = 1 & & \forall s \in S \\
& \sigma^{\mathcal{D}}_{v,s}(r) \geq 0 \ \forall r \in R_{v,s}, s \in S & &
\end{aligned}
$$

4 Finding the Best Patrolling Strategy

We now focus on the problem of finding the best patrolling strategy given that we know (from Sect. 3.3) the best signal–response strategy for each vertex v in which $\mathcal{D}$ can place. Given the current vertex of $\mathcal{D}$ and the sequence of the last, say n, vertices visited by $\mathcal{D}$ (where n is a tradeoff between effectiveness of the solution and computational effort), a patrolling strategy is usually defined as a randomization over the next adjacent vertices [2]. We define $v^* = \arg\min_{v \in V}\{g_v\}$, where g_v is the value returned by the optimization problem described in Sect. 3.3, as the vertex that guarantees the maximum expected utility to $\mathcal{D}$ over all the SRG–vs. We show that the maxmin equilibrium strategy in PG prescribes that $\mathcal{D}$ places at v^*, waits for a signal, and responds to it.

Theorem 4. *Without false positives and missed detections, if* $\forall t \in T$ *we have that* $|S(t)| \geq 1$*, then any patrolling strategy is dominated by the placement in* v^**.*

Proof. Any patrolling strategy different from the placement in v^* should necessarily visit a vertex $v' \neq v^*$. Since the alarm system is not affected by missed detections, every attack will raise a signal which, in turn, will raise a response yielding a utility of g_x where x is the current position of $\mathcal{D}$ at the moment of the attack. Since $\mathcal{A}$ can observe the current position of $\mathcal{D}$ before attacking, $x = \arg\max_{v \in P}\{g_v\}$ where P is the set of the vertices patrolled by $\mathcal{D}$. Obviously, for any $P \supseteq \{v^*\}$ we would have that $g_x \geq g_{v^*}$ and therefore placing at v^* and waiting for signal is the best strategy for $\mathcal{D}$. □

4.1 Computing the Best Placement

Under the absence of false positives and missed detections, Theorem 4 simplifies the computation of the patrolling strategy by reducing it to the problem of finding v^*. To such aim, we must solve SRG–v for each possible starting vertex v and select the one with maximum expected utility for $\mathcal{D}$. Since all the vertices are possible starting points, we should face this difficult problem (see Theorem 1) $|V|$ times, computing, for each signal, the covering routes from all the vertices. To avoid this issue, we ask whether there exists an algorithm that in the worst case allows us to consider a number of iterations, such that solving the problem for a given node v could help us finding the solution for another node v'. So, considering a specific set of targets, we wonder whether a solution for COV–SET with starting vertex v can be used to derive, in polynomial time, a solution to COV–SET for another starting vertex v'. To answer this question, we need to encode an instance of COV–SET in a different way, embedding the selection of the starting node in the structure of the graph. More precisely, we represent an instance of COV–SET $I = \langle G = (V, E), T, d, v\rangle$ with the equivalent instance $I' = \langle G' = (V', E'), c', T, d', \hat{v}\rangle$ defined in the following way:

- $V' = V \cup \{\hat{v}\}$, $E' = E \cup \{(\hat{v}, v_i), \forall v_i \in V\}$
- c' is a weight function such that $c(e) = 1$ if $e \in E \cup \{(\hat{v}, v\})$ and $c(e) = M$ otherwise (M is a big constant);
- $d'(t) = d(t) + 1$, $\forall t \in T$.

With $\hat{v}$ we denote the dummy vertex that is always the starting node. We highlight the fact that, under this new encoding scheme, changing the starting vertex translates to rewriting the weights of c. Following the approach of [7], we can show that even the locally modified version of this problem, where a single weight is updated, is hard.

Definition 6. LM–COV–ROUTE *(Locally modified)*
INSTANCE: a graph $G = (V, E)$, a set of targets T with penetration times d, two weight functions c_1 and c_2 that coincide except for one edge, and a covering route r_1 such that, under c_1, $T(r_1) = T$.
QUESTION: is T a covering set under c_2?

Theorem 5. LM*–COV–ROUTE is $\mathcal{NP}$–complete.*

Proof. Let us consider the Restricted Hamiltonian Circuit problem (RHC) which is known to be $\mathcal{NP}$–complete. RHC is defined as follows: given a graph $G_H = (V_H, E_H)$ and an Hamiltonian path $P = \{h_1, \ldots, h_n\}$ for G_H such that $h_i \in V_H$ and $(h_1, h_n) \notin E_H$, find an Hamiltonian circuit for G_H. From such instance of RHC, following the approach of [7], we build the following instance for LM–COV–ROUTE:

- $V = T = V_H \cup \{v_s, v_t\}$;
- $E = \{(v_s, h_1), (v_s, h_{n-1})\} \cup \{(v_t, u) | (u, h_{n-1}) \in E_H\} \cup \overline{E_H}$ where $\overline{E_H}$ is the complete set of edges obtained by augmenting E_H;
- $d(v_s) = 0$, $d(v_t) = n + 1$, $d(t) = n$ for any $t \in T$;
- $c_1(e) = 1$ if $e \in E \cup \{(v_s, h_1), (v_s, h_{n-1})\} \cup \{(v_t, u) | (u, h_{n-1}) \in E_H\}$, $c_1(e) = (1 + \epsilon)$ otherwise (for any $\epsilon > 0$);
- $c_1 = c_2$ except for $c_2(v_s, h_1) = 1 + \epsilon$;
- $r_1 = \langle v_s, h_1, \cdots, h_n, v_t \rangle$.

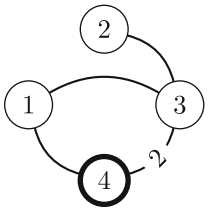
It is easy to verify that G_H admits a Hamiltonian circuit if and only if T admits a covering route under c_2. □

This shows that iteratively applying Algorithm 1 to SRG–v for each starting vertex v is the best we can do.

4.2 Robustness to Missed Detections

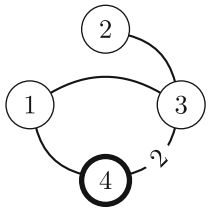
A deeper analysis of Theorem 4 can show that its scope does include cases where missed detections are present up to a non–negligible extent. For such cases, placement–based strategies keep being optimal even in the case when the alarm systems fails in detecting an attack. We encode the occurrence of this robustness property in the following proposition, which we shall prove by a series of examples.

Proposition 1. *There exist Patrolling Games where staying in a vertex, waiting for a signal, and responding to it is the optimal patrolling strategy for $\mathcal{D}$ even with a missed detection rate $\alpha = 0.5$.*



t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$
1	0.5	1	1
2	0.5	3	1
3	0.5	2	1
4	0.5	2	1

(a) Equation 1 holds for $\alpha \leq 0.25$



t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$
1	1	1	1
2	1	3	1
3	1	2	1
4	1	2	1

(b) Equation 1 holds for $\alpha \leq 0.5$

Fig. 1. Two examples proving Proposition 1.

Proof. The expected utility for $\mathcal{D}$ given by the placement in v^* is $(1-\alpha)(1-g_{v^*})$, where $(1-\alpha)$ is the probability with which the alarm system correctly generates a signal upon an attack and $(1-g_{v^*})$ denotes $\mathcal{D}$'s payoff when placed in v^*. A non–placement–based patrolling strategy will prescribe, by definition, to move between at least two vertices. From this simple consideration, we observe that an upper bound to the expected utility of any non–placement strategy is entailed by the case where $\mathcal{D}$ alternately patrols vertices v^* and v_2^*, where v_2^* is the second best vertex in which $\mathcal{D}$ can statically place. Such scenario give us an upper bound over the expected utility of non–placement strategies, namely $1-g_{v_2^*}$. It follows that a sufficient condition for the placement in v^* being optimal is given by the following inequality:

$$(1-\alpha)(1-g_{v^*}) > (1-g_{v_2^*}) \tag{1}$$

To prove Proposition 1, it then suffices to provide a Patrolling Game instance where Eq. 1 holds under some non–null missed detection rate α. In Fig. 1(a) and (b), we report two of such examples. The depicted settings have unitary edges except where explicitly indicated. For both, without missed detections, the best patrolling strategy is a placement $v^* = 4$. When allowing missed detections, in Fig. 1(a) it holds that $g_{v^*} = 0$ and $g_{v_2^*} = 0.75$, where $v^* = 4$ and $v_2^* = 1$. Thus, by Eq. 1, placement $v^* = 4$ is the optimal strategy for $\alpha \leq 0.25$. Under the same reasoning scheme, in Fig. 1(b) we have that $g_{v^*} = 0$ and $g_{v_2^*} = 0.5$, making the placement $v^* = 4$ optimal for any $\alpha \leq 0.5$. □

5 Experimental Evaluation

We evaluate the scalability of Algorithm 1 and the quality of the solution returned by our heuristic algorithm for a set of instances of SRG–v. We do not include results on the evaluation of the algorithm to solve completely a PG, given that it trivially requires asymptotically $|V|$ times the effort required by the resolution of a single instance of SRG–v.

Testbed. In real deployment scenarios, the model parameters should be derived from the particular features that characterize the particular setting one must deal with. Besides the graph topology, which depends on the environment, target values and deadlines can be derived from available statistics or manually

assigned by domain experts. The need of such process to derive model parameters makes building a large dataset of realistic instances not an easy task. In fact, such task would deserve a separate treatment by its own. On the other side, by means of a preliminary experimental evaluation, we observed how completely random instances are very likely of being not significant. Indeed, very frequently the variance of the compute time among completely random generated instances is excessively large. For these reasons, we decided to use a random generator where some parameters can be fixed while others are randomly chosen. We restricted our attention to basic, but significant, instances with all–targets graphs, arc costs set to 1, penetration times to $|T(s)|-1$, and the number of arcs is drawn from a normal distribution with mean ϵ, said edge density and defined as $\epsilon = |E|/\frac{|T(s)|(|T(s)|-1)}{2}$ (other parameters are randomly generated from uniform distributions, unless otherwise specified). Instances constructed with such mechanism include hard ones since the existence of a covering route over $T(s)$ would imply the existence of an Hamiltonian path on the graph. We explore two parameter dimensions: the number of targets $|T|$ and ϵ. Algorithms are developed in MATLAB and run on a 2.33 GHz LINUX machine.

Exact Algorithm Scalability. Table 1 shows the total compute time required to solve instances with a single signal, that can be generated by any target under attack. Table 2 refers to instances with multiple signals, where the targets covered by a signal and the probability that a target triggers a signal are randomly chosen according to a uniform distribution (in this second table $|T|$ is fixed to 16). Values are averages over 100 random instances and give insights on the computation effort along the considered dimensions. The results show that the problem is computationally challenging even for a small number of targets and signals.

Table 1. Compute times (in seconds) for single–signal instances.

ϵ \ $\lvert T\rvert$	6	8	10	12	14	16	18
.25	0.07	0.34	1.91	11.54	82.26	439.92	4068.8
.5	0.07	0.38	4.04	53.14	536.7	4545.4	$\geq$ 5000
.75	0.09	0.96	11.99	114.3	935.74	7276.62	$\geq$ 7000
1	0.14	1.86	17.46	143.05	1073.19	7964.49	$\geq$ 8000

Table 2. Compute times (in seconds) for multi–signal instances.

m \ $\lvert T(s)\rvert$	5	10	15
2	-	17.83	510.61
3	-	33	769.3
4	0.55	35.35	1066.76
5	0.72	52.43	1373.32

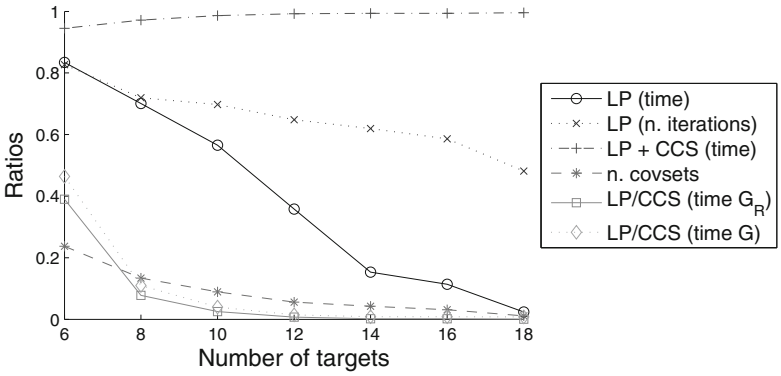


Fig. 2. Ratios evaluating dominances.

Figure 2 shows the impact of discarding dominated actions from the game. It depicts the trend of some performance ratios for different metrics. We shall call $\mathcal{G}$ the complete game including all $\mathcal{D}$'s dominated actions and $\mathcal{G}_R$ the reduced game; CCS will denote the full version of Algorithm 1 and LP will denote the linear program to solve SRG–v. Each instance has edge density $\epsilon = .25$ and is solved for a random starting vertex v; we report average ratios for 100 instances. "n. covsets" is the ratio between the number of covering sets in $\mathcal{G}_R$ and in $\mathcal{G}$. Dominated actions constitute a large percentage, increasing with the number of targets. This result indicates that the structure of the problem has some redundancy. LP times (iterations) report the ratio between $\mathcal{G}_R$ and $\mathcal{G}$ for the time (iterations) required to solve the minmax linear program. A relative gain directly proportional to the percentage of dominated covering sets is observable (LP has less variables and constraints). A similar trend is not visible when considering the same ratio for the total time which includes CCS. Indeed, the time needed by CCS largely exceed LP's and removal of dominated actions determines a polynomial additional cost which can be seen in the slightly increasing trend of the curve. The relative gap between LP and CCS compute times can be assessed by looking at the LP/CCS curve: when more targets are considered the time taken by LP is negligible w.r.t. CCS's. This shows that removing dominated actions is useful, allowing a small improvement in the average case, and assuring an exponential improvement in the worst case.

Heuristic Solution Quality. Figure 3 reports the performance of the heuristic algorithm (here we set $\delta = 2$) in terms of $\mathcal{D}$'s expected utility ratio $(1 - g_v)/(1 - \hat{g}_v)$, where g_v is the expected utility of $\mathcal{A}$ at the equilibrium considering all the covering sets and $\hat{g}_v$ is the expected utility of $\mathcal{A}$ at the equilibrium when covering sets are generated by our heuristic algorithm. The performance of our heuristic algorithm is well characterized by ϵ, providing fairly good approximations for $\epsilon > 0.25$, the ratio going to 1 as $|T|$ increases, because there are more edges and, consequently, there is a higher probability for the heuristics to find longer routes. The figure suggests that assessing the membership of our problem to the $\mathcal{APX}$ class could be an interesting problem.

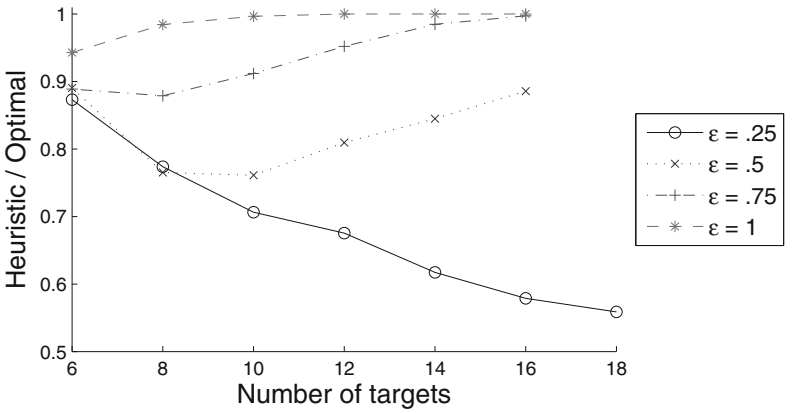


Fig. 3. Optimal vs heuristic algorithms.

6 Conclusions and Future Research

In this paper, to the best of our knowledge, we provide the first Security Game for large environments surveillance, e.g. for wildlife protection, that can exploit an alarm system with spatially uncertain signals. We propose a simple model of alarm systems that can be widely adopted with every specific technology and we include it in the state–of–art patrolling models obtaining a new security game model. We show that the problem of finding the best patrolling strategy to respond to a given alarm signal is $\mathcal{FNP}$–hard even when the game is zero sum. Then, we provide an exponential–time exact algorithm to find the best patrolling strategy to respond to a given alarm signal. We provide also a heuristic algorithm returning approximate solutions to deal with very large game instances. Furthermore, we show that if every target is alarmed and no missed detections are present, then the best patrolling strategy prescribes that the patroller stays in a given place waiting for a alarm signal. We show that such a strategy may be optimal even for missed detection rates up to 50 %. Finally, we experimentally evaluate our algorithms in terms of scalability (for the exact algorithm) and approximation ratio (for the heuristic algorithm).

In future works, we shall study the membership (or not) of our problem to $\mathcal{APX}$ class, design approximation algorithms with theoretical guarantees and investigate the impact of missed detections and false positives.

References

1. An, B., Brown, M., Vorobeychik, Y., Tambe, M.: Security games with surveillance cost and optimal timing of attack execution. In: AAMAS, pp. 223–230 (2013)
2. Basilico, N., Gatti, N., Amigoni, F.: Patrolling security games: definition and algorithms for solving large instances with single patroller and single intruder. Artif. Intell. **184–185**, 78–123 (2012)

3. Basilico, N., Gatti, N., Rossi, T.: Capturing augmented sensing capabilities and intrusion delay in patrolling-intrusion games. In: IEEE Symposium on Computational Intelligence and Games, CIG 2009, pp. 186–193, September 2009
4. Basilico, N., Carpin, S., Chung, T.: Distributed online patrolling with multi-agent teams of sentinels and searchers. In: DARS (2014)
5. Bellare, M., Goldwasser, S.: The complexity of decision versus search. SIAM J. Comput. **23**(1), 97–119 (1994)
6. Blum, A., Haghtalab, N., Procaccia, A.D.: Lazy defenders are almost optimal against diligent attackers. In: AAAI, pp. 573–579 (2014)
7. Böckenhauer, H.J., Forlizzi, L., Hromkovič, J., Kneis, J., Kupke, J., Proietti, G., Widmayer, P.: Reusing optimal tsp solutions for locally modified input instances. In: IFIP TCS, pp. 251–270 (2006)
8. Munoz de Cote, E., Stranders, R., Basilico, N., Gatti, N., Jennings, N.: Introducing alarms in adversarial patrolling games. In: AAMAS, pp. 1275–1276 (2013)
9. Delle Fave, F.M., Jiang, A., Yin, Z., Zhang, C., Tambe, M., Kraus, S., Sullivan, J.P.: Game-theoretic patrolling with dynamic execution uncertainty and a case study on a real transit system. JAIR **50**, 321–367 (2014)
10. Ford, B.J., Kar, D., Fave, F.M.D., Yang, R., Tambe, M.: PAWS: adaptive game-theoretic patrolling for wildlife protection. In: International conference on Autonomous Agents and Multi-Agent Systems, AAMAS 2014, Paris, France, 5–9 May 2014, pp. 1641–1642 (2014)
11. Garcia-Sanchez, A.J., Garcia-Sanchez, F., Garcia-Haro, J.: Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops. Comput. Electron. Agric. **75**(2), 288–303 (2011)
12. Jain, M., An, B., Tambe, M.: An overview of recent application trends at the AAMAS conference: security, sustainability, and safety. AI Mag. **33**(3), 14–28 (2012)
13. Jain, M., Tsai, J., Pita, J., Kiekintveld, C., Rathi, S., Tambe, M., Ordóñez, F.: Software assistants for randomized patrol planning for the lax airport police and the federal air marshal service. Interfaces **40**(4), 267–290 (2010)
14. Ko, B.C., Park, J.O., Nam, J.Y.: Spatiotemporal bag-of-features for early wildfire smoke detection. Image Vis. Comput. **31**(10), 786–795 (2013)
15. Osborne, M.J.: An Introduction to Game Theory, vol. 3. Oxford University Press, New York (2004)
16. Paruchuri, P., Tambe, M., Ordóñez, F., Kraus, S.: Security in multiagent systems by policy randomization. In: AAMAS, pp. 273–280 (2006)
17. Savelsbergh, M.W.: Local search in routing problems with time windows. Ann. Oper. Res. **4**(1), 285–305 (1985)
18. Von Stengel, B., Zamir, S.: Leadership with commitment to mixed strategies (2004)
19. Vorobeychik, Y., An, B., Tambe, M., Singh, S.P.: Computing solutions in infinite-horizon discounted adversarial patrolling games. In: Proceedings of the Twenty-Fourth International Conference on Automated Planning and Scheduling, ICAPS 2014, Portsmouth, New Hampshire, USA, 21–26 June 2014 (2014)
20. Yang, R., Ford, B., Tambe, M., Lemieux, A.: Adaptive resource allocation for wildlife protection against illegal poachers. In: International Conference on Autonomous Agents and Multiagent Systems (AAMAS) (2014)
21. Yick, J., Mukherjee, B., Ghosal, D.: Wireless sensor network survey. Comput. Netw. **52**(12), 2292–2330 (2008)