# Coordinating Multiple Defensive Resources in Patrolling Games with Alarm Systems

Nicola Basilico
University of Milan
Via Comelico, 39/41
Milano, Italy
nicola.basilico@unimi.it

Andrea Celli
Politecnico di Milano
Piazza Leonardo da Vinci, 32
Milano, Italy
andrea.celli@polimi.it

Giuseppe De Nittis
Politecnico di Milano
Piazza Leonardo da Vinci, 32
Milano, Italy
giuseppe.denittis@polimi.it

Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci, 32
Milano, Italy
nicola.gatti@polimi.it

## ABSTRACT

Alarm systems represent a novel issue in Security Games, requiring new models that explicitly describe the dynamic interaction between the players. Recent works studied their employment, even considering various forms of uncertainty, and showed that disregarding them can lead to arbitrarily poor strategies. One of the key problems is computing the best strategy to respond to alarm signals for each mobile defensive resource. The current literature only solves the basic single–resource version of such problem. In this paper, we provide a solution for the multi–resource case addressing the challenge of designing algorithms to coordinate a scaling–up number of resources. First, we focus on finding the minimum number of resources assuring non–null protection to every target. Then, we deal with the computation of multi–resource strategies with different degrees of coordination among resources resorting to adversarial team game models. For each considered problem, we provide algorithms and their theoretical and empirical analysis.

## Keywords

Security Games, Alarm System, Multiple Defensive Resources

## 1. INTRODUCTION

Security Games are a successful application of non–cooperative game theory in the real world [14]. In the mainstream approach, a security scenario is modeled with a 2–player game, between a *Defender* and an *Attacker* and the best strategy of the Defender is derived according to the Stackelberg paradigm [19]. Upon such basic formulation, literature studied issues like resource scheduling constraints [15], bounded rationality [20], Attacker's observation models [2], and protection of infrastructures [8].

Real security systems typically exploit *sensors* triggering *alarms* when attacks are detected but suffering from various forms of uncertainty. Mobile defensive resources (a.k.a. *patrollers*) can respond to alarm signals covering the targets potentially under attack. Disregarding alarm signals can lead to strategies arbitrarily worse than those obtained when alarm signals are exploited [3]. Nevertheless, the study of how to include alarm systems in Security Games is largely unexplored and represents a challenging open problem. In particular, the central question is: *given an alarm signal, how should the Defender respond to it at best?* To tackle this problem, we need to adopt models that explicitly describe a dynamic interaction between the players. Notice that such models are richer than the ones without alarms, commonly studied in literature, which only consider the static placement of mobile resources. In [3], the authors study the scenario with only one resource available to the Defender and with sensors affected by *spatial uncertainty*, i.e., an alarm signal is raised with any attack, but the Defender is uncertain on the actual attacked location, as, e.g., in border patrolling [1]. In such situations, the best strategy is to stay in a location, wait for an alarm signal and then respond to it at best. This last task is proved FNP–hard even in tree graphs. The work proposed in [5] studies the scenario with sensors suffering both from *spatial uncertainty* and *false negatives* when only one resource is available to the Defender, showing that it is PSPACE–hard. This work exposes an interesting relation between missed detection rates and optimal strategies. For small missed detection rates, placement–based strategies (which are optimal in absence of missed detections and prescribe to place in a location, wait for an alarm signal and then respond to it [4]) keep being optimal for the Defender. As the missed detection rate increases, placement–based strategies are outperformed by cyclically patrolling a number of targets while waiting for a signal to respond to. This shows that, in situations where the missed detection rate is small, assuming no false negatives is not a limiting assumption.

**Original contributions**. We focus on settings with a spatially uncertain alarm system and multiple defensive resources. The challenge is designing algorithms able to scale up with the number of resources. The problem of finding the best Defender's strategy when a number of resources are given is trivially FNP–hard from the case with a single resource. We show that even the problem of finding the minimum number of resources assuring non–null protection to every target, which is of high relevance in practice due to resource costs and to the need for assuring a minimum level of protection to each target, is log–APX–complete on arbitrary graphs, while it is in FP in tree and cycle graphs, usually representing docks and borders respectively. Then, we study the problem of finding the best strategy to respond to any alarm signal once an allocation of resources in the environment is given, according to dif-

**Appears in:** *Proc. of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

ferent *degrees of coordination* among the resources, each described by an adversarial team game with different forms of strategies (*correlated* or *mixed*). To the best of our knowledge, finding equilibria in adversarial team games is a largely unexplored problem and it may be crucial when the goal is to coordinate different (e.g., defensive) resources in a strategic situation. We provide exact and approximation algorithms and show they perform very well empirically, scaling up to more than 100 targets and 15 resources both in correlated and mixed strategies with a negligible optimality gap.

## 2. SECURITY GAME MODEL AND PREVIOUS RESULTS

Our security game is a generalization of [4], obtained allowing the Defender to control an arbitrary number of resources, denoted by $m$, instead of just a single one. We summarize its main features. A *patrolling setting* is a graph $G = (V, E)$ representing an environment where areas that can be attacked are given by $n$ target vertices, denoted as $T \subseteq V$. A target $t \in T$ has a value $\pi(t) \in (0, 1]$ and requires $d(t) \in \mathbb{N}^+$ time units (*penetration time*) to be compromised. Edges in $E$ have unitary cost, while $\omega^*_{i,j}$ is the smallest traveling cost between vertices $i$ and $j$. An *alarm system* $(S, p)$ generates a signal $s \in S$ if target $t$ is attacked with probability $p(s \mid t)$. $S$, $p$ and any generated signal $s$ is common knowledge. We call $T(s) = \{t \in T \mid p(s \mid t) > 0\}$ the *support of a signal* $s$ and $S(t) = \{s \in S \mid p(s \mid t) > 0\}$ the *support of a target* $t$. Given the novelty of the setting, we start from the basic case where $p(\varnothing \mid t) = p(s \mid \varnothing) = 0$ for any $t \in T$ and $s \in S$, i.e., we assume no false positives or missed detections. The results we derive under such assumption are functional for addressing the general case (that we shall address in future works) and can be effectively applied in situations where a small false negatives rate is present (see [5]). We also assume that $|S(t)| \geqslant 1$ for any $t \in T$.

A 2–player security game takes place between an Attacker $\mathcal{A}$ and a Defender $\mathcal{D}$. In this game, $\mathcal{A}$ seeks to gain value by compromising some target while $\mathcal{D}$ can control $m$ mobile resources by specifying a movement strategy for them. At any turn of the game, $\mathcal{A}$ and $\mathcal{D}$ play simultaneously: $\mathcal{A}$ observes the position of the $m$ resources and decides whether to attack a target—we assume that $\mathcal{A}$ can instantly reach the attacked target, this can be relaxed as shown in [6]—or to wait. On the opposite side, $\mathcal{D}$ has no information about $\mathcal{A}$ but, if an attack is present, it observes the alarm signal and decides where to move the $m$ resources along the graph. If $\mathcal{A}$ attacks target $t$ and a resource traverses $t$ in any of the next $d(t)$ turns, then $\mathcal{D}$ and $\mathcal{A}$ receive payoffs of 1 and 0, respectively. Otherwise, they receive $1 - \pi(t)$ and $\pi(t)$.

Given a resource $j$ and a signal $s$, an arbitrary finite length $l$ sequence $r_{s,i} = (r(0), r(1), \ldots, r(l))$ where $r(0)$ is any vertex and $r(i)$ is any target in $T(s)$ is a *route* for resource $j$ under signal $s$. A route can be instantiated to a graph walk for resource $j$ starting from $r(0)$ and traveling any shortest path between $r(i)$ and $r(i+1)$. For any $i \in \{1, \ldots, l\}$, call $A(r(i)) = \sum_{h=0}^{i-1} \omega^*_{r(h),r(h+1)}$ (the traveling cost required by such walk for reaching $r(i)$). We say that $r_{s,j}$ is a *covering route* if $\forall i \in \{1, \ldots, l\}$, we have that $A(r(i)) \leqslant d(r(i))$. Any other target $t$ not appearing in the route is not visited or visited after $d(t)$ time units from the start of the route. Covering routes are abstract representations for $\mathcal{D}$'s pure strategies when a signal $s$ is raised. When a resource $j$ follows a covering route, all and only the targets appearing in such route are protected (notice that this representation is without loss of information). A *joint* covering route is an $m$–tuple $r_s = \langle r_{s,1}, r_{s,2}, \ldots, r_{s,m} \rangle$. To simplify the notation, we will omit signal $s$ when clear from the context.

The resolution approach for $m = 1$ is given in [4] where it is shown how with no false positives and missed detections, the best strategy is to place the resource on a vertex $v$ and, when a signal $s$ is received execute a *signal response strategy* which, in general, is defined as a mixed strategy over all the covering routes starting from $v$. Denoted with $g_v$ $\mathcal{D}$'s expected utility when responding to signals from $v$, then the best strategy is to place the resource on the vertex maximizing such value. This best placement keeps being the optimal strategy even when small missed detection rates are present [5]. For the case of multiple resources, we can derive an analogous result.

**Theorem 1** *Without false positives and missed detections, if $\forall t \in T$ it holds that $|S(t)| \geqslant 1$, then the optimal strategy prescribes to place each resource on some vertex and execute a signal response from there.*

*Proof Sketch.* Call $P^*$ the set of $m$ vertices specifying the optimal placement of each resource. Denote with $g^*_P$ the value, in terms of expected utility for $\mathcal{A}$, of the resulting signal response game, that is the strategic game where $\mathcal{A}$ has to choose a target to attack and $\mathcal{D}$ must choose a joint covering route where each resource starts from the associated vertex in $P^*$. Consider any patrolling strategy that, in absence of signals, prescribes to visit at least one vertex $v' \notin P^*$. Since the alarm system is not affected by missed detections, an attack will always raise a signal which, in turn, will raise a response yielding a utility of $g_X$, where $X$ is the set of vertices corresponding to the current positions of the resources at the moment of the attack. Since $\mathcal{A}$ can observe the position of the resources before attacking, $X = \arg\max_{P \in \Pi}\{g_P\}$, where $\Pi$ is the space of possible joint locations of the resources given the patrolling strategy. Since we assumed that $P^*$ is optimal, for any $X \in \Pi$ we would have that $g_X \geqslant g^*_P$. Thus, in absence of signals, $\mathcal{D}$ has no strict incentive to move the resources away from $P^*$. □

This result allows us to work under the same problem decomposition operated in the single–resource case which rewrites the game as two independent sub–games. The first is the *Signal Response Game* where $\mathcal{D}$ has to determine how to respond when receiving any signal from any given joint location resource. As we will show, solving such game amounts to finding a strategy that, upon reception of a signal, draws and executes a covering route for any resource deployed in the environment. The second sub–game is the *Patrolling Game* where the joint placement for the $m$ resources must be determined. Such placement must encode, for each resource, its starting position and must be selected taking into account the expected utility $\mathcal{D}$ gets when playing the associated Signal Response Games (where the starting positions for the resources are specified by such joint placement). The literature results for the single–resource case show how, despite the game being constant–sum, its resolution is FNP–hard even in tree graphs, its difficulty mainly being in generating the covering routes under signal $s$ and starting from $v$. An approximation algorithm for such difficult subproblem is given in [3].

The multi–resource setting clearly inherits this hardness profile, posing the major need for algorithms capable of providing acceptable solutions in affordable time and showing scalability w.r.t. the number of resources in the game. The presence of multiple resources also poses the problem of determining their minimum number in order to guarantee non-null protection to every target. Last but not least, coordination becomes an issue that, as we will show, has a critical role during signal response. We start from the minimum number of resources problem and then we deal with the game resolution under different coordination schemes.

## 3. MINIMIZING THE NUMBER OF RESOURCES

Ideally, the best number of defensive resources depends on both the level of protection that can be achieved and on the costs of the resources. Realistic scenarios often pose the requirement that, for each target $t \in T$, there is at least a resource in a vertex $v$ such that $\omega^*_{v,t} \leqslant d(t)$, i.e., it can cover $t$ by $d(t)$, thus stopping a potential ongoing attack. Indeed, the authority in charge of the security system's deployment usually requires some protection guarantees over all the targets, even if it forces a non–optimal placement. In other words, leaving a target completely uncovered is generally excluded *a priori*. This entails the existence of, and the need of computing, a minimum number of resources necessary for the protection of any environment. An upper bound can be defined too as the number of resources such that, for any signal $s$, there is a response strategy to $s$ covering all the targets in $T(s)$ thus assuring the Defender a utility of 1. We deal with this problem by introducing the concept of covering placement and subsequently finding the minimum covering placement.

**Definition 1 (Covering Placement)** *A covering placement is a set $P = \{p_1, \dots, p_m\}$ where $p_i \in V$, $p_i \neq p_j$ if $i \neq j$ and for any $t \in T$ it holds that $\omega^*_{p_i,t} \leqslant d(t)$ for some $p_i \in P$.*

### 3.1 Arbitrary instances

With arbitrary graphs, we state the following results.

**Theorem 2** *Computing the minimum covering placement is log–APX–hard even in the basic case where all the vertices are targets with penetration times equal to 1.*

*Proof.* Log–APX–hardness of our problem directly follows from the optimization version of DOMINATING–SET [11] that is known to be log–APX–complete. DOMINATING–SET is defined as follows.

**Definition 2** *The optimization version of the DOMINATING–SET problem is defined as:*

- *INPUT: a graph $\overline{G} = (\overline{V}, \overline{E})$;*
- *OUTPUT: $\overline{V}' \subseteq \overline{V}$ such that $|\overline{V}'|$ is minimized under the constraint that for all $v \in \overline{V} \backslash \overline{V}'$ there is at least a $v' \in \overline{V}'$ such that $(v, v') \in \overline{E}$.*

The mapping between the two problems is:

- $V = \overline{V}$;
- $E = \overline{E}$;
- $T = V$;
- $d(t) = 1$ for every $t \in T$;
- $\pi(t) = 1$ any for every $t \in T$;
- $S = \{s_1\}$ with $p(s_1 \mid t) = 1$ for every $t \in T$.

The objective function $m$ of the problem of minimizing the covering placement equals the objective function $|\overline{V}|$ of DOMINATING–SET. Therefore any $\alpha$–approximation of the minimum covering placement is an $\alpha$–approximation of DOMINATING–SET. □

**Theorem 3** *Computing the minimum covering placement is in log–APX.*

*Proof.* The membership to log–APX follows from the fact that every instance of our problem can be directly formulated as an optimization version of a SET–COVER instance and that SET–COVER is in log–APX [10]. Let us notice that our problem is more general than DOMINATING–SET, justifying the need for considering SET–COVER, which is defined as follows.

**Definition 3** *The optimization version of the SET–COVER problem is defined as:*

- *INPUT: universe set $U$, family $\mathcal{L} = \{L \subseteq U\}$;*
- *OUTPUT: a cover $\mathcal{C} = \{L \in \mathcal{L}\}$ of universe set $U$ such that $|\mathcal{C}|$ is minimum.*

The mapping between the two problems is:

- $U = T$;
- $\mathcal{L} = \{L_v : v \in V\}$ where $L_v = \{t : t \in T \text{ and } \omega^*_{v,t} \leqslant d(t)\}$.

Notice that $\omega^*_{v,t} \leqslant d(t)$ can be computed in polynomial time and therefore the mapping can be performed in polynomial time. The objective function $|\mathcal{C}|$ of SET–COVER equals the objective function $m$ of the problem of minimizing the covering placement. Therefore any $\alpha$–approximation of SET–COVER is an $\alpha$–approximation of the minimum covering placement. □

The proof of Theorem 3 shows that we can find a solution to our problem by formulating it as a SET–COVER and then by using algorithms for this latter problem: Integer Linear Programming (ILP) for finding the exact solution and greedy or local–search algorithms to find an approximate solution, see [10] or [16], respectively.

### 3.2 Special instances: tree and cycle graphs

Interestingly, with particular instances that are rather common in real–world applications the problem is optimally solvable in polynomial time. Let us start from the following lemma.

**Algorithm 1** TreePlacements($v$)

```
1: $res_v \leftarrow 0$ for every $v \in V$
2: if $v$ is a leaf then
3:     return $(\infty, d(v) - 1)$
4: else
5:     for all $v' \in Succ(v)$ do
6:         $\big(Cov(v'), UnCov(v')\big) \leftarrow$ TreePlacements$\big(v'\big)$
7:     if $\min_{v'}\{UnCov(v'), d(v)\} - \min_{v'} Cov(v') \geqslant 0$ then
8:         return $(\min_{v'} Cov(v') + 1, \infty)$
9:     else if $\min_{v'}\{UnCov(v'), d(v)\} - 1 \geqslant 0$ then
10:        return $(\infty, \min_{v'}\{UnCov(v'), d(v)\} - 1)$
11:    else
12:        $res_v \leftarrow 1$
13:        return $(1, \infty)$
```

**Lemma 4** *A minimum covering placement in a tree rooted in $\hat{v}$ can be computed in polynomial time with Algorithm 1.*

*TreePlacements* (Algorithm 1) works recursively taking as input a vertex $v \in V$. To ease its description, let us assume that $T = V$. The case including non–target vertices only amounts to minor modifications. Binary variables $res_v$ are initially set to 0 and their assertion corresponds to place a resource on $v \in V$. With $Succ(v) \subseteq V$ we denote all the immediate successors of $v$ on the path leaving the root. The idea is to recursively allocate resources by processing the graph in a bottom–up fashion, from its leaves to the root. Let us consider a function call for a generic vertex $v$. By recursively invoking *TreePlacements*$(v')$ for each $v' \in Succ(v)$ we get, for each successor, a *coverage profile* defined with a pair of values $\big(Cov(v'), UnCov(v')\big)$. They encode the following conditions under the currently built resource placement. If variables $res_v$ of the subtree rooted in $v'$ constitute a covering placement for the whole subtree, the coverage profile is such that $Cov(v') = k < \infty$ where $k$ is the distance between $v'$ and the closest resource on such subtree and $UnCov(v') = \infty$. Otherwise, the coverage profile is such that $Cov(v') = \infty$ and $UnCov(v') = k < \infty$ where $k$ is the distance from $v'$ by which we need to place a resource to have a covering placement for the subtree rooted in $v'$.

We start from an empty placement and derive coverage profiles recursively from the base case in which $v$ is a leaf (Line 2). Since $d(v) \geqslant 1$ and costs are unitary, a resource on a leaf is never necessary: we can always cover it from any ancestor whose distance from $v$ is $\leqslant d(v) - 1$. Hence the coverage profile for the base case is $(\infty, d(v) - 1)$ (Line 3).

Let us consider the recursive step in which $v$ is a non–leaf vertex. From Line 7 we have all the coverage profiles of each one of the immediate successors of $v$. We must return the coverage profile of $v$ and decide if a resource must be placed on it. We have three cases.

*Case 1 (Lines 7–8)*: the subtree rooted at $v$ is covered by the resources we placed in such subtree. Hence we do not need any resource in $v$. Moreover, the ancestor of $v$ will be at distance $\min_{v'} Cov(v') + 1$, where $v' \in Succ(v)$, from the closest of such resources.

*Case 2 (Lines 9–10)*: the subtree rooted at $v$ is *not* covered by the resources we placed in such subtree. We can achieve coverage by placing a resource in $v$ or in any ancestor whose distance from $v$ should not exceed $\min_{v'}\{UnCov(v'), d(v)\} - 1$, where $v' \in Succ(v)$. Since we are trying to minimize the number of resources, there is always interest in postponing a resource allocation in our bottom–up processing of the tree. Therefore, we do not allocate a resource in $v$ and we return the coverage profile naturally resulting from the above considerations (Line 10).

*Case 3 (Lines 12–13)*: no resource in any ancestor of $v$ can complete the coverage for the subtree rooted at $v$. We are forced to place a resource in $v$ which makes the associated coverage profile equal to $(1, \infty)$ (Line 13).

Algorithm 1 can be adopted also to solve cycle graphs by extracting the $n$ linear subgraphs spanning all the $n$ targets, solving each of them with Algorithm 1, and then selecting the solution with the least number of resources. We summarize the above positive results in the following theorem.

**Theorem 5** *The problem of finding the minimum covering placement in trees and cycle graphs is in* $\mathsf{FP}$.

## 4. SIGNAL RESPONSE

We focus on computing a signal–response strategy for $m$ resources given a joint placement $P \subseteq V^m$. Notice that we do not assume it to be exactly the minimum one but only to satisfy the lower bound discussed in the previous section. W.l.o.g., we assume that only one signal $s$ is present and that $T(s) = T$, omitting $s$ in our formulas. (The general case comes by refining notation.)

Any resource $i$ will always move along a covering route. We denote by $R_i$ the set of covering routes for resource $i$ starting from $p_i$ (we will omit the dependency on $p_i$ since $P$ is always fixed in the scope of a signal response game). Covering routes can be computed exactly or approximately by means of the methods proposed in [3]. The presence of multiple resources poses the problem of their coordination [7]. We consider three different coordination schemes for which we define three *Signal Response Oracles* (SROs). Each oracle works on a different adversarial team game and returns the signal response strategy from a given joint placement $P$ under the corresponding scheme. Strategies for $\mathcal{D}$ and $\mathcal{A}$ are denoted as $\sigma^{\mathcal{D}}$ and $\sigma^{\mathcal{A}}$, respectively. If not defined differently, $\sigma^{\mathcal{A}} : T \to [0, 1]$ gives the probability $\sigma^{\mathcal{A}}(t)$ of attacking $t$.

### 4.1 Full coordination SRO (FC–SRO)

We assume that $\mathcal{D}$ acts as central planner and executor of the signal–response strategy defined on the joint moves of the resources. Equivalently, the defensive resources are players of a constant–sum adversarial team game in which they play correlated strategies. This scheme captures scenarios where resources are connected to a central control unit from which orders are issued (e.g., police patrols equipped with radio transceivers). Formally, we have $\sigma^{\mathcal{D}} : R \to [0, 1]$, $R = \times_{i=1}^{m} R_i$ and $\sigma^{\mathcal{D}}(r)$ is the probability of playing joint covering route $r$. We define $I(r, t)$ as a function returning 1 if and only if target $t$ is protected by $r$ (with notation overload, we allow $r$ to be both a joint and a single–resource covering route) and 0 otherwise. For any $r \in R$ and $t \in T$, $(r, t)$ is a game outcome where $\mathcal{A}$ and $\mathcal{D}$ receive $U_{\mathcal{A}}(r, t) = (1 - I(r, t))\pi(t)$ and $U_{\mathcal{D}}(r, t) = 1 - U_{\mathcal{A}}(r, t)$, respectively. The game can be solved computing the maxmin strategy by linear programming. However, each $R_i$ can have exponential cardinality and its computation entails the resolution of an $\mathsf{NP}$–hard problem. Even adopting approximation methods and working with incomplete sets of covering routes, the set of joint covering routes is exponential in the number of resources. Nevertheless, we observe that in our case there is always a maxmin equilibrium where $\mathcal{D}$ plays at most $|T|$ routes with strictly positive probability as proved in [17], showing that working with the complete set of all the joint covering routes is unnecessary.

For this reason, we devise a row–generation approach that iteratively generates rows in the game matrix (corresponding to joint covering routes). The row generation routine first generates, for each resource $i$, a set of covering routes $R_i$. Then, it considers a joint covering route $r'$, sets $R = \{r'\}$ and finds the $\mathcal{A}$'s minmax strategy in the corresponding constant–sum game using $R$ as the action space for $\mathcal{D}$. Subsequently, given the $\mathcal{A}$'s strategy, the following ILP is solved to find the $\mathcal{D}$'s best response among all the joint covering routes without their explicit enumeration:

$$
\begin{aligned}
\max \quad & 1 - \sum_{t \in T} \sigma^{\mathcal{A}}(t)\pi(t)(1 - y_t) \quad \text{s.t.} \\
& \sum_{i=1}^{m} \sum_{r \in R_i} I(r, t) x_{ir} - y_t \geqslant 0 \quad \forall t \in T \\
& \sum_{r \in R_i} x_{ir} = 1 \quad \forall i \in \{1 \dots m\} \\
& y_t \in \{0, 1\} \quad \forall t \in T \\
& x_{ir} \in \{0, 1\} \quad \forall i \in \{1 \dots m\}, r \in R_i
\end{aligned}
$$

where $x_{ir}$ is a binary variable taking value of 1 when route $r \in R_i$ is selected for resource $i$, $y_t$ is a binary variable taking value of 1 when target $t$ is protected by the set of selected covering routes ($y_t$ can be safely relaxed on $[0, 1]$, turning the ILP into a MILP), and $\sigma^{\mathcal{A}}$ is $\mathcal{A}$'s minmax strategy in the previously solved game. From the solution of this problem we obtain a joint covering route

$\hat{r}$ where resource $i$ plays route $r$ if $x_{ir} = 1$. Clearly, $\hat{r}$ is a best response for $\mathcal{D}$ in the incumbent game equilibrium. If $\hat{r} \notin R$, then it is included and the game is solved again, otherwise the set of actions played by $\mathcal{D}$ at the equilibrium is found. However, finding the best response cannot be done in polynomial time unless $\mathsf{P} = \mathsf{NP}$, as a consequence of the following theorem.

**Theorem 6** *Given $m$ resources, a set of single–resource covering routes $R_i$ for each resource $i$, and a subset of targets $T'$, deciding if there is at least a joint route covering all the targets in $T'$ is NP–hard.*

*Proof.* NP–hardness of our problem follows from a direct reduction from the decision version of SET–COVER (see Definition 3). The mapping between the two problems is:

- $m = k$;
- $V = U \cup \{v\}$ where $v$ is the starting vertex;
- graph $G$ is fully connected;
- $T' = U$;
- for any $L \in \mathcal{L}$, we have a route $r$ covering all the targets $t \in L$;
- $R_i = \mathcal{L}$ for every resource $i$;
- $d(t) = |T'|$ for every $t \in T$;
- $\pi(t) = 1$ for every $t \in T$;
- $S = \{s_1\}$ with $p(s_1 \mid t) = 1$ for every $t \in T$.

There is at least a joint route covering all the targets $T'$ if and only if there is at least a cover $\mathcal{C}$ with $|\mathcal{C}| \leqslant k$. □

Nevertheless, the problem admits a polynomial–time algorithm with constant approximation, returning a joint covering route providing an expected utility of al least 0.63 $OPT_{BR}$ where $OPT_{BR}$ is the expected utility of the best response.

**Theorem 7** *Given the attacker's strategy, $m$ resources and a set of single–resource covering routes $R_i$ for each resource $i$, the problem of maximizing the defender's utility admits a polynomial–time algorithm with approximation guarantee of $1 - \frac{1}{e} \simeq 0.63$ where $e$ is the Euler's constant.*

*Proof.* We show this by exploiting a well-known result from submodular optimization of set functions. We start from some basic definitions.

**Definition 4** *A set function $f : 2^U \to \mathbb{R}$ is* submodular *if, for every $A, B \subseteq U$, it holds that $f(A \cap B) + f(A \cup B) \leqslant f(A) + f(B)$.*

**Definition 5** *A set function $f : 2^U \to \mathbb{R}$ is monotone if, for every $A \subseteq B \subseteq U$, it holds that $f(A) \leqslant f(B)$.*

**Definition 6** *A* matroid *is a pair $(U, I)$ such that $U$ is a finite set and $I \subseteq 2^U$ is a collection of subsets of $U$ satisfying the following two properties:*

1. $\forall B \in I, \forall A \subset B \Rightarrow A \in I$;
2. $\forall A, B \in I, |A| < |B| \Rightarrow \exists e \in B \backslash A$ *such that* $A \cup \{e\} \in I$.

**Definition 7** *A partition matroid $(U, I)$ is a matroid where $U$ is partitioned into disjoint sets $U_1, U_2, \ldots, U_l$ with associated integers $k_1, k_2, \ldots, k_l$, and sets $I$ are defined as $I = \{X \subseteq U : |X \cap U_i| \leqslant k_i, \forall i = 1, \ldots, l\}$.*

**Definition 8** *Given a monotone submodular set function $f : 2^U \to \mathbb{R}_+$ and a partition matroid $(U, I)$, call MON–SUBMODULAR–PART–MAT the problem defined as defined as $\max_{H \in I} f(H)$.*

The objective function (to be maximized) is: $1 - \sum_{t \in T} \sigma^A(t)\pi(t)(1 - y_t)$ where $\sigma^A$ is the strategy of $A$, $\pi(t)$ is the value of target $t$ and $y_t$ is a binary variable which has value 1 if and only if target $t$ is covered. This is equivalent to maximizing the value of covered targets, defined as $\sum_{t \in T} \sigma^A(t)\pi(t)y_t$.

Let $U = \bigcup_{i=1}^{m} R_i$ be the ground set of the union of all possible covering routes. We consider the function $f : 2^U \to \mathbb{R}_+$ defined as $f(r) = \sum_{t \in T} \sigma^A(t)\pi(t)y_t$, where $r$ is a joint covering route $r = \langle r_1, \ldots, r_m \rangle$ and $y_t$ is a binary variable taking value 1 if and only if target $t$ is covered by at least one (single–resource) covering route $r_i$.

We show that function $f$ is monotone submodular.

- $f$ is monotone since all its discrete derivatives are nonnegative, i.e., for every $X \subseteq U$ and $e \in U$ it holds that $\Delta(e|X) \geqslant 0$.
- The submodularity property is satisfied since $X \subseteq Y \subseteq U$ implies that $Y$ covers at least all the targets covered by $X$ and thus the marginal utility of adding an element $e \in U \backslash Y$ is surely bigger when $e$ is added to $X$.

We define the partition matroid $(U, I)$ as follows:

- the ground set $U$ is defined as $U = \bigcup_{i=1}^{m} \bigcup_{r \in R_i} \langle i, r \rangle$, describing the set of couples composed of player and covering route.
- the ground set $U$ is partitioned as $U_1, \ldots, U_m$ where $\forall i \in \{1, \ldots, m\}$, $U_i = \bigcup_{r \in R_i} \langle i, r \rangle$.
- the sets $I$ are defined as $I = \{X \subseteq U : |X \cap U_i| \leqslant 1, \forall i = 1, \ldots, m\}$.

With this formulation, our optimization problem can be seen as a MON–SUBMODULAR–PART–MAT with function $f$ over the partition matroid $(U, I)$. There is a polynomial–time approximation algorithm based on non–oblivious local search [12] with an approximation guarantee of $1 - \frac{1}{e} \simeq 0.63$. □

## 4.2 Partial coordination SRO (PC–SRO)

Partial coordination models those situations where $\mathcal{D}$ can act as central planner but cannot communicate with each resource to prescribe a joint action. Equivalently, the defensive resources are players of a constant–sum adversarial team game in which they play mixed strategies [18]. Real scenarios falling in this scope can be characterized by resources for which a communication with the control unit is not available (e.g., patrolling units operating in stealth mode). Formally, we define a $(m + 1)$–player game where resource players $\mathcal{D}_1, \ldots, \mathcal{D}_m$ compete together against $\mathcal{A}$. Each resource strategy is defined as $\sigma_i^{\mathcal{D}} : R_i \to [0, 1]$, where $\sigma_i^{\mathcal{D}}(r_i)$ denotes the probability of having resource $i$ following covering route $r_i \in R_i$. A game outcome is again defined with $(r, t)$—$r$ is a joint covering route—where each $\mathcal{D}_i$ receives the same utility $U_{\mathcal{D}}$ as previously defined.

The proper solution concept of this game is the team maxmin equilibrium [18] whose equilibrium constraints are non–linear non–convex. The minmax/maxmin value with 3 or more players may be irrational and not exactly computable even when the payoffs can assume only three different integer values [13]. Approximating the minmax value of 2 players against 1 player within an additive error of $\frac{1}{3z^2}$ where $z$ is the number of actions of each player is NP–hard even with payoffs $\{0, 1\}$ [9]. The work proposed in [13] provides a quasi–polynomial algorithm with complexity $O(z^{l\frac{\log(z)}{\epsilon^2}})$ approximating with additive error $\epsilon$ the minmax value with $l$ players with payoffs in $[0, 1]$. The algorithm is not applicable in our case even for toy instances and non–negligible $\epsilon$ (e.g., with 10 actions, 2 resources, and $\epsilon = 0.5$, the number of needed iterations is of the order of $10^{18}$). Another crucial issue is that the size of $\mathcal{D}$'s payoff matrix increases exponentially in the number of resources. However, by exploiting the structure of the problem, we can provide a Non–Linear mathematical Program (NLP) whose size is linear in the number of resources compressing exponentially the size of the game (notice that such a compression is not possible for FC–SRO):

$$\min_{\sigma_1^{\mathcal{D}} \dots \sigma_m^{\mathcal{D}}} v \quad \text{s.t.}$$

$$v - \pi(t) \prod_{i=1}^{m} \big(1 - \sum_{r \in R_i} I(r,t)\sigma_i^{\mathcal{D}}(r)\big) \geqslant 0 \quad \forall t \in T$$

$$\sum_{r \in R_i} \sigma_i^{\mathcal{D}}(r) = 1 \quad \forall i \in \{1 \dots m\}$$

$$\sigma_i^{\mathcal{D}}(r) \geqslant 0 \quad \forall i \in \{1 \dots m\}, r \in R_i$$

Our objective is the minimization of $v$, i.e., the expected utility of $\mathcal{A}$, w.r.t. the strategies played by the resources of $\mathcal{D}$. We observe that such strategies are implemented independently by the resources.

Constraints (2) are the core of the formulation. Let $I : R \times T \rightarrow \{0, 1\}$ be the following indicator function:

$$I(r,t) = \begin{cases} 1 & \text{if } t \in r \\ 0 & \text{otherwise} \end{cases}.$$

For each target $t$, we require that $v$ is greater or equal than the protection probability the Defender can ensure to $t$. We compute such protection probability multiplying the value $\pi(t)$ of target $t$ by the product of the non–covering probabilities of each resource $i$ of the Defender. Specifically, given a resource $i$ and a target $t$, such non–covering probability is expressed as $1 - \sum_{r \in R_i} I(r,t)\sigma_i^{\mathcal{D}}(r)$, where $\sum_{r \in R_i} I(r,t)\sigma_i^{\mathcal{D}}(r)$ is the covering probability of $t$ given by $i$ when executing routes $r$ with probability $\sigma_i^{\mathcal{D}}(r)$. We observe that these constraints provide an exponential compression of the size of the game such that the resulting game is linear in the number of resources.

Constraints (3), (4) require that the pure strategies of each resource are feasible. In other words, for each resource $i$, we require that each action $r$ is played with non–negative probability $\sigma_i^{\mathcal{D}}(r)$ and the sum of such probabilities is equal to 1. We solve the program by means of global–optimization techniques, which are largely unexplored in the equilibrium computation field, able to find the global maximum within a given accuracy and, in the case the posed time limit is expired, able to return the quality of the best solution found w.r.t. the tightest upper bound found. Although global optimization may require exponential time, we will show that in our problem it provides satisfactory performances.

### 4.3 No coordination SRO (NC–SRO)

When no coordination is allowed among resources, we are ruling out not only strategy correlation but also joint planning. In this scenario, resources are unaware of each other and act as single players against $\mathcal{A}$. Defensive resources are completely non–coordinated and $\mathcal{A}$ can observe the strategy of each resource and play at best by taking into account all of them. This case is modeled with $m$ independent single resource signal response games—as defined in [4]—where the game associated with resource $i$ has $p_i$ as starting vertex and is played on the restricted set of targets $T_i = \{t \in T \mid \omega_{p_i,t} \leqslant d(t)\}$. For each game $i$, the maxmin strategy is computed taking $R_i$ as the action space of $\mathcal{D}_i$. Thus, given the $m$ resource allocation strategies obtained in this way, we can compute the game value as $1 - \max_{t \in T} \prod_{i=1}^{m} \big(1 - \sum_{r_i \in R_i} \sigma_i^{\mathcal{D}}(r_i) I(r,t)\big)\pi(t)$.

## 5. OVERALL RESOLUTION APPROACH

The resolution approach we use is sketched in Fig. 1. We remark that such approach can be adopted when the Defender controls *any* number of resources, which are at least the ones required for the minimum covering placement. For the sake of simplicity, here we consider the case with the minimum number of resources as defined in Sec. 3 (we will keep this assumption also for the experimental results shown in Sec. 6). We start by tackling the problem of finding the minimum–size resource placement by solving the associated SET–COVER formulation (or our polynomial algorithm with trees and cycles). If the problem cannot be solved exactly in a short time, we adopt the greedy approximation algorithm of [10] and then apply a simple local search [16] to improve the greedy solution. This gives us a number of resources $m$ assuring a (sub)minimal covering placement. As previously stated, in absence of false negatives and false positives, the best strategy when no signal is raised is to statically place the $m$ resources in the best covering placement in terms of expected utility maximization. To deal with this, we resort again to a simple variation of the local search procedure to enumerate covering placements of exactly $m$ resources in an anytime fashion. For each considered covering placement, we compute sets $R_i$ for each resource $i$ as mentioned in the previous section and then we run the signal response oracles we introduced.

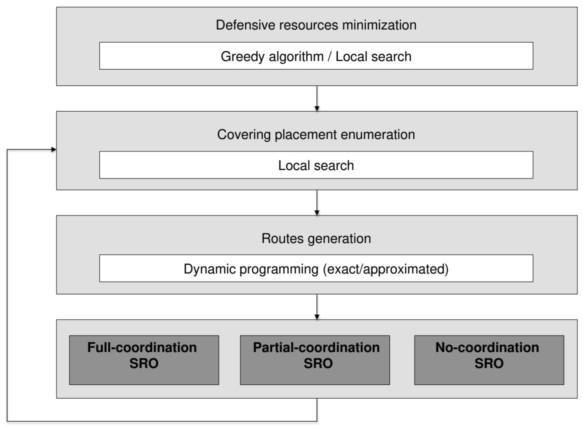

Defensive resources minimization
Greedy algorithm / Local search
Covering placement enumeration
Local search
Routes generation
Dynamic programming (exact/approximated)
Full-coordination SRO
Partial-coordination SRO
No-coordination SRO

**Figure 1: Overview of the proposed resolution approach.**

## 6. EXPERIMENTAL EVALUATIONS

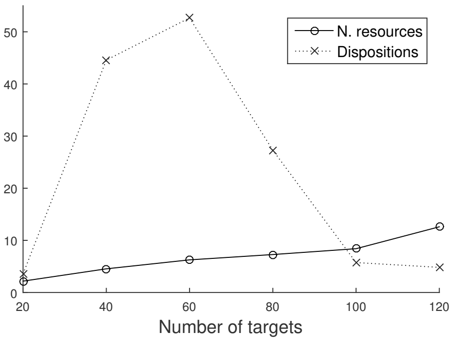
To evaluate our algorithms, we implemented a random instance generator that leverages some domain knowledge we gathered by discussing with domain experts from the Italian National Police.

The patrolling instances randomly generated could represent realistic urban environments, such as streets, squares or districts, capturing scenarios with police patrolling units placed in police stations. In the graphs, all the vertices are targets, $|T| \in \{20, 40, 60, 80, 100, 120\}$, edges have unitary costs, the average indegree of each vertex is 3, and $d(t)$ is: $d(t) = 3$ for $|T| \in \{20, 40\}$, $d(t) = 4$ for $|T| \in \{60, 80\}$ and $d(t) = 5$ for $|T| \in \{100, 120\}$; $\pi(t)$ is drawn uniformly from $(0, 1]$. There is one single signal covering all the targets, corresponding to the worst case for the computation of the routes (the problem of scaling up with exponential signals is still open). In our experiments, oracles were run with a deadline of 60 minutes. All the numerical results are averaged on 50 instances for each $|T|$. For the instances employed here, we used the exact method of [3] to generate the covering routes, requiring a compute time comparable to the approximation algorithm. We use GUROBI to solve exactly LPs and (M)ILPs, BARON to solve exactly NLPs, and Python for the algorithms. We run experiments on a UNIX computer with 2.3GHz CPU and 128GB RAM.

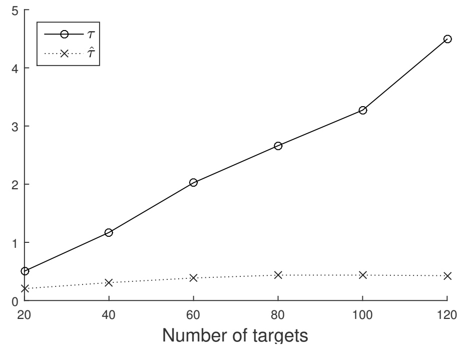
**Resources and allocations** We initially evaluate the performance of the algorithms to find the best covering placement. The ILP–based method finding the optimal solution scales up to instances with 500 vertices when a time limit of 1 hour is used. The greedy algorithm returning an approximate solution achieves an average optimality gap $< 5\%$ with up to 500 targets requiring a negligible compute time, suggesting that it can provide good suboptimal results even for settings with more than 500 targets. In the following analysis, we use the ILP–based method.

First, we focus on the characteristics exhibited by our experimental settings in terms of number of resources needed for the covering placement and the degree of overlap over the targets covered by the resources, see Fig. 2. We observe that the average number of used resources is linear in the number of targets (see Fig. 2(a)), which is reasonable in real–world scenarios. To quantify the overlap degree induced by a given covering placement $P = \{p_1, p_2, \ldots, p_m\}$, we define two complementary indicators. We recall that $T_i$ is the subset of targets that can be covered (reached by their $d(t)$) from vertex $p_i$. Then the following quantity counts the number of *extra coverings* induced by $P$: $\eta = \sum_{i=1}^{m} |T_i| - |T|$. Notice that $0 \leqslant \eta \leqslant (|T| - m)(m - 1)$ reaches its maximum value when each resource covers a different subset of $|T| - m + 1$ targets. We denote with $\tau = \frac{\eta}{|T|}$ the average overlap per target and $\hat{\tau} = \frac{\eta}{(|T|-m)(m-1)}$ the normalized overlap. Fig. 2(b) shows how these two indicators vary with the number of targets. The index $\tau$ grows linearly in the number of targets due to the linear growth of the number of resources while $\hat{\tau}$ has a slower growth since for any resource $i$ the number of covered targets $|T_i|$ is usually not as big as $|T|$.

Now we focus on the covering placement enumeration phase and we evaluate the number of covering placements that our algorithm is able to consider within a 60 minutes deadline. Referring again to Fig. 1, notice that each additionally considered placement requires to compute the covering routes sets and to run one of the three SROs. To find an upper bound over the number of covering placements we use the NC–SRO since it is the oracle requiring the minimum compute time. The number of generated covering placements is reported in Fig. 2(a). The curve grows reaching its maximum for $|T| = 60$ and then decreases. Indeed, when the size of the problem is small, the algorithm terminates before the deadline, returning few placements. On the other side, when the size of the problem is large, due to the time limit and the large amount of time required by the computation of the covering routes, the time to compute new placements lowers significantly and thus the curve decreases. We remark that we are able to solve large instances up to 120 targets.



(a) Resources and dispositions

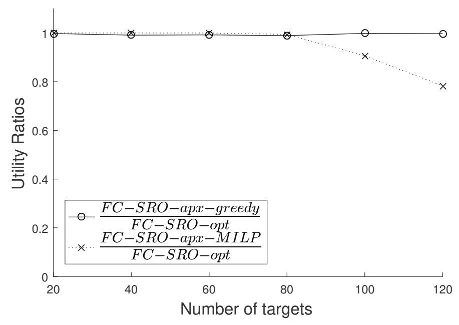


(b) Overlap degree

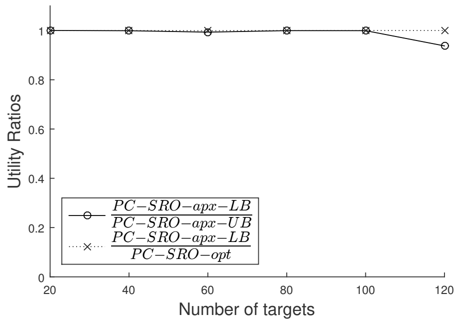
**Figure 2: Analysis of the instances before an attack.**

**SROs quality and performance** We compare the performance of the three SROs, both in terms of utility and compute time. We call FC–SRO–opt the FC oracle using MILP for the row generation, FC–SRO–apx–MILP the same with a time limit of 1 hour, FC–SRO–apx–greedy the FC oracle using the approximation algorithm for the row generation with a time limit of 1 hour. Furthermore, we call PC–SRO–opt the PC oracle, PC–SRO–apx–LB the lower bound returned by the PC oracle with a time limit of 1 hour and PC–SRO–apx–UB the upper bound returned by the PC oracle with a time limit of 1 hour. In Fig. 3(a), we report, as the number of targets varies, the average ratio between the utilities returned by FC–SRO–apx–greedy and FC–SRO–opt and the average ratio between the utilities returned by FC–SRO–apx–MILP and FC–SRO–opt. FC–SRO–apx–MILP returns the optimal solution when $|T| \leqslant 60$. Interestingly, FC–SRO–apx–greedy provides solutions very close to the optimal ones (with an efficiency always larger than 99%) and dramatically outperforms FC–SRO–apx–MILP with 80 targets or more. In Fig. 3(b), we report, as the number of targets varies, the average ratio between the utilities returned by PC–SRO–apx–LB and PC–SRO–opt and the average ratio between the utilities returned by PC–SRO–apx–LB and PC–SRO–apx–UB. PC–SRO–apx–LB returns the optimal solution when $|T| \leqslant 40$. Also in this case, the approximation algorithm (i.e., PC–SRO–apx–LB) provides performance very close to the optimal solution (with an efficiency always larger than 99%).

Fig. 4(a) shows the comparison between the different oracles. Partial coordination introduces negligible inefficiency in patrolling games (this does not hold in generic games). Surprisingly, PC–SRO–apx–LB is better than FC–SRO–apx–greedy for 100 targets and therefore it should be considered as approximation oracle for full coordination. This results is due to the fact that FC–SRO–

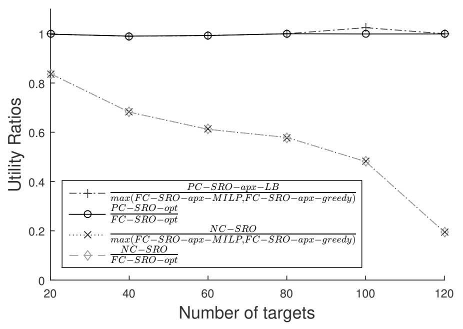


(a) FC–SRO

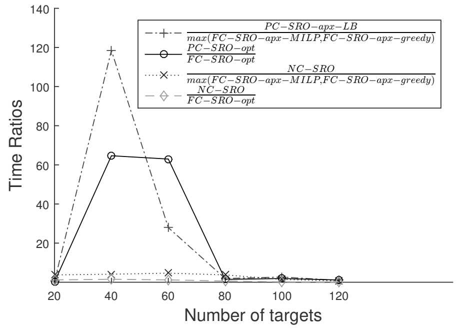


(b) PC–SRO

**Figure 3: FC–SRO and PC–SRO evaluation.**



(a) Utility ratios



(b) Time ratios

**Figure 4: Utility and time ratios of the three SROs.**

apx–greedy cannot finish its execution within the timeout we set. No coordination is extremely inefficient w.r.t. full and partial coordination. Fig. 4(b) shows the time ratio between the different oracles. Interestingly, the full coordination oracles are much faster than the partial coordination oracles when $|T| \leqslant 60$. Beyond 60 targets, FC–SRO–apx–greedy/MILP and PC–SRO–apx–LB stop at the time limit of 1 hour and so their compute time is the same.

**Utility trend in time** Finally, we focus on the evolution of the utility in time, after different placements have been enumerated and evaluated. Fig. 5 shows two instances with $|T| = 60$ where we compare the performances when using the three different oracles with a 20 minutes time limit: we use FC–SRO–apx–greedy for full coordination and PC–SRO–apx–LB for partial coordination. While NC–SRO is quite constant even though several placements are evaluated, FC–SRO and PC–SRO utilities increase, with the former always preceding the latter, being faster in finding good solutions.

## 7. CONCLUSIONS AND FUTURE WORKS

In this work, we study a security game with the presence of a spatially uncertain alarm system and we address the novel generalization towards settings in which the Defender can control multiple mobile resources. Solving this problem requires new models that explicitly describe the interaction between the players. We propose a scalable resolution approach for dealing with the new algorithmic problems that such generalization introduces.

Future research will involve adapting our algorithms to cases in which the number of resources available to the Defender is greater than the one required for the minimum covering placement. In addition, we plan to work on the model by allowing the presence of false positives and missed detection in the alarm systems as well as the presence of multiple resources on the attacker side.

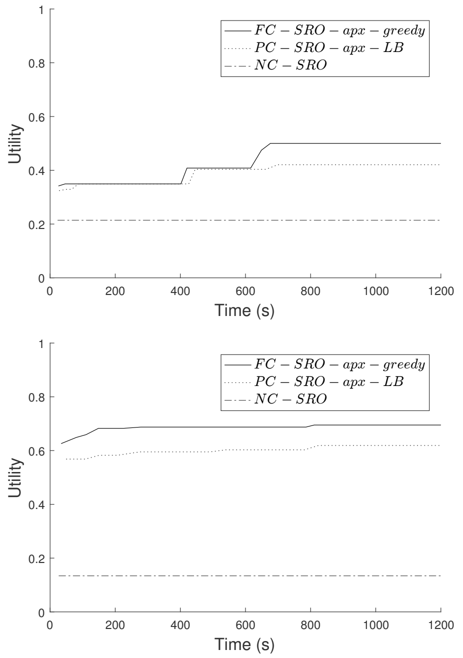


**Figure 5: Utility trend in time.**

## REFERENCES

[1] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, 2008.

[2] B. An, M. Brown, Y. Vorobeychik, and M. Tambe. Security games with surveillance cost and optimal timing of attack execution. In *AAMAS*, pages 223–230, 2013.

[3] N. Basilico, G. De Nittis, and N. Gatti. Adversarial patrolling with spatially uncertain alarm signals. *arXiv:1506.02850*, 2015.

[4] N. Basilico, G. De Nittis, and N. Gatti. A security game model for environment protection in the presence of an alarm system. In *GameSec*, pages 192–207. 2015.

[5] N. Basilico, G. De Nittis, and N. Gatti. A security game combining patrolling and alarm-triggered responses under spatial and detection uncertainties. In *AAAI*, pages 397–403, 2016.

[6] N. Basilico, N. Gatti, and T. Rossi. Capturing augmented sensing capabilities and intrusion delay in patrolling-intrusion games. In *CIG*, pages 186–193, 2009.

[7] N. Basilico, N. Gatti, and F. Villa. Asynchronous multi-robot patrolling against intrusion in arbitrary topologies. In *AAAI*, pages 1224–1229, 2010.

[8] A. Blum, N. Haghtalab, and A. D. Procaccia. Lazy defenders are almost optimal against diligent attackers. In *AAAI*, pages 573–579, 2014.

[9] C. Borgs, J. T. Chayes, N. Immorlica, A. T. Kalai, V. S. Mirrokni, and C. H. Papadimitriou. The myth of the Folk Theorem. *GAME ECON BEHAV*, 70(1):34–43, 2010.

[10] V. Chvatal. A greedy heuristic for the set-covering problem. *MATH OPER RES*, 4(3):233–235, 1979.

[11] B. Escoffier and V. T. Paschos. Completeness in approximation classes beyond apx. *THEOR COMPUT SCI*, 359(1):369–377, 2006.

[12] Y. Filmus and J. Ward. The power of local search: Maximum coverage over a matroid. In *STACS*, volume 14, pages 601–612, 2012.

[13] K. A. Hansen, T. D. Hansen, P. B. Miltersen, and T. B. Sørensen. Approximability and parameterized complexity of minmax values. In *WINE*, pages 684–695, 2008.

[14] M. Jain, B. An, and M. Tambe. An overview of recent application trends at the AAMAS conference: Security, sustainability, and safety. *AI MAG*, 33(3):14–28, 2012.

[15] 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.

[16] N. Musliu. Local search algorithm for unicost set covering problem. In *IEA/AIE*, 2006.

[17] L. S. Shapley and R. N. Snow. Basic solutions of discrete games. *ANN MATH STUD*, 24:27–35, 1950.

[18] B. von Stengel and D. Koller. Team-maxmin equilibria. *GAME ECON BEHAV*, 21(1):309–321, 1997.

[19] B. Von Stengel and S. Zamir. Leadership with commitment to mixed strategies. Technical report, 2004.

[20] R. Yang, C. Kiekintveld, F. Ordóñez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. In *AAAI*, pages 458–464, 2011.