Artificial Intelligence 246 (2017) 220–257

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Adversarial patrolling with spatially uncertain alarm signals

Nicola Basilico [a,*], Giuseppe De Nittis [b], Nicola Gatti [b]

[a] Department of Computer Science, University of Milan, Milano, Italy
[b] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Milano, Italy

ARTICLE INFO

Article history:
Received 15 June 2015
Received in revised form 6 August 2016
Accepted 26 February 2017
Available online 4 March 2017

Keywords:
Security games
Adversarial patrolling
Algorithmic game theory

ABSTRACT

When securing complex infrastructures or large environments, constant surveillance of every area is not affordable. To cope with this issue, a common countermeasure is the usage of cheap but wide-ranged sensors, able to detect suspicious events that occur in large areas, supporting patrollers to improve the effectiveness of their strategies. However, such sensors are commonly affected by uncertainty. In the present paper, we focus on spatially uncertain alarm signals. That is, the alarm system is able to *detect* an attack but it is uncertain on the *exact position* where the attack is taking place. This is common when the area to be secured is wide, such as in border patrolling and fair site surveillance. We propose, to the best of our knowledge, the first Patrolling Security Game where a *Defender* is supported by a spatially uncertain *alarm system*, which non-deterministically generates *signals* once a target is under attack. We show that finding the optimal strategy is FNP-hard even in tree graphs and APX-hard in arbitrary graphs. We provide two (exponential time) exact algorithms and two (polynomial time) approximation algorithms. Finally, we show that, 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, which, unfortunately, affect every commercial alarm system. We evaluate our methods in simulation, assessing both quantitative and qualitative aspects.

1. Introduction

Security Games model the task of protecting physical environments as a non-cooperative game between a *Defender* and an *Attacker* [1]. These games usually take place under a *Stackelberg* (a.k.a. *leader–follower*) paradigm [2], where the Defender (*leader*) commits to a strategy and the Attacker (*follower*) first observes such commitment and then best responds to it. As discussed in the seminal work [3], finding a leader–follower equilibrium is computationally tractable in games with one follower and complete information, while it becomes hard in Bayesian games with different types of Attacker. The availability of such computationally tractable aspects of Security Games led to the development of algorithms capable of scaling up to large problems, making them deployable in the security enforcing systems of several real-world applications. The first notable examples are the deployment of police checkpoints at the Los Angels International Airport [4] and the scheduling of federal air marshals over the U.S. domestic airline flights [5]. More recent case studies include the positioning of U.S. Coast Guard patrols to secure crowded places, bridges, and ferries [6] and the arrangement of city guards to stop fare evasion in Los Angeles Metro [7]. Finally, a similar approach is being tested and evaluated in Uganda, Africa, for the

* Corresponding author.
E-mail address: nicola.basilico@unimi.it (N. Basilico).

http://dx.doi.org/10.1016/j.artint.2017.02.007
0004-3702/

protection of wildlife [8]. Thus, Security Games emerged as an interesting game theoretical tool and then showed their on-the-field effectiveness in a number of real security scenarios.

We focus on a specific class of security games, called *Patrolling Security Games*. These games are modeled as infinite-horizon extensive-form games in which the Defender controls one or more *patrollers* moving within an environment, represented as a finite graph. The Attacker, besides having knowledge of the strategy to which the Defender committed to, can observe the movements of the patrollers at any time and use such information in deciding the most convenient time and target location to attack [9]. When multiple patrollers are available, coordinating them at best is in general a hard task which, besides computational aspects, must also keep into account communication issues [10]. However, the patrolling problem is tractable, even with multiple patrollers, in border security (e.g., line and cycle graphs), when patrollers have homogeneous moving and sensing capabilities and all the vertices composing the border share the same features [11]. Scaling this model involved the study of how to compute patrolling strategies in scenarios where the Attacker is allowed to perform multiple attacks [12]. Similarly, coordination strategies among multiple defenders are investigated in [13]. In [14], the authors study the case in which there is a temporal discount on the targets. Extensions are discussed in [15], where coordination strategies between defenders are explored, in [16], where a resource can cover multiple targets, and in [17] where attacks can be detected at different stages with different associated utilities. Finally, some theoretical results about properties of specific patrolling settings are provided in [18]. In the present paper, we provide a new model of Patrolling Security Games in which the Defender is supported by an *alarm system* deployed in the environment.

1.1. Motivating scenarios

Often, in large environments, a constant surveillance of every area is not affordable while focused inspections triggered by alarms are more convenient. Real-world applications include UAVs surveillance of large infrastructures [19], wildfires detection with CCD cameras [20], agricultural fields monitoring [21], surveillance based on wireless sensor networks [22], and border patrolling [23]. Alarm systems are in practice affected by *detection* uncertainty, e.g., missed detections and false positives, and *localization* (a.k.a. spatial) uncertainty, e.g., the alarm system is uncertain about the exact target under attack. We summarily describe two practical security problems that can be ascribed to this category. We report them as examples, presenting features and requirements that our model can properly deal with. In the rest of the paper we will necessarily take a general stance, but we encourage the reader to keep in mind these two cases as reference applications for a real deployment of our model.

1.1.1. Fight to illegal poaching

Poaching is a widespread environmental crime that causes the endangerment of wildlife in several regions of the world. Its devastating impact makes the development of surveillance techniques to contrast this kind of activities one of the most important matters in national and international debates. Poaching typically takes place over vast and savage areas, making it costly and ineffective to solely rely on persistent patrol by ranger squads. To overcome this issue, recent developments have focused on providing rangers with environmental monitoring systems to better plan their inspections, concentrating them in areas with large likelihood of spotting a crime. Such systems include the use of UAVs flying over the area, alarmed fences, and on-the-field sensors trying to recognize anomalous activities.[1] In all these cases, technologies are meant to work as an alarm system: once the illegal activity is recognized, a signal is sent to the rangers base station from where a response is undertaken. In the great majority of cases, a signal corresponds to a spatially uncertain localization of the illegal activity. For example, a camera-equipped UAV can spot the presence of a pickup in a forbidden area but cannot derive the actual location to which poachers are moving. In the same way, alarmed fences and sensors can only transmit the location of violated entrances or forbidden passages. In all these cases a signal implies a restricted, yet not precise, localization of the poaching activity. The use of Security Games in this particular domain is not new (see, for example, [8]). However, our model allows the computation of alarm response strategies for a given alarm system deployed on the field. This can be done by adopting a discretization of the environment, where each target corresponds to a sector, values are related to the expected population of animals in that sector, and deadlines represent the expected completion time of illegal hunts (these parameters can be derived from data, as discussed in [8]).

1.1.2. Safety of fair sites

Fairs are large public events attended by thousands of visitors, where the problem of guaranteeing safety for the hosting facilities can be very hard. For example, Expo 2015, the recent Universal Exposition hosted in Milan, Italy, saw an average of about 100,000 visits per day. This poses the need for carefully addressing safety risks, which can also derive from planned act of vandalism or terrorist attacks. Besides security guards patrols, fair sites are often endowed with locally installed monitoring systems. Expo 2015 employed around 200 baffle gates and 400 metal detectors at the entrance of the site. The internal area was constantly monitored by 4000 surveillance cameras and by 700 guards. Likely, when one or more of these devices/personnel identified a security breach, a signal was sent to the control room together with a circumscribed request of intervention. This approach is required because, especially in this kind of environments, detecting a security breach

[1] See, for example, http://wildlandsecurity.org/.

and neutralizing it are very different tasks. The latter one, in particular, usually requires a greater effort involving special equipment and personnel whose employment on a demand basis is much more convenient. Moreover, the detecting location of a threat is in many cases different from the location where it could be neutralized, making the request of intervention spatially uncertain. For instance, consider a security guard or a surveillance camera detecting the visitors' reactions to a shooting rampage performed by some attacker. In examples like these, we can restrict the area where the security breach happened but no precise information about the location can be gathered since the attacker will probably have moved. Our model could be applied to provide a policy with which schedule interventions upon a security breach is detected in some particular section of the site. In such case, targets could correspond to buildings or other installations where visitors can go. Values and deadlines can be chosen according to the importance of targets, their expected crowding, and the required response priority.

1.2. Alarms and security games

While the problem of managing a sensor network to optimally guard security-critical infrastructures has been investigated in restricted domains, e.g. [24], the problem of integrating alarm signals together with adversarial patrolling is almost completely unexplored. The only work that can be classified under this scope is [25]. The paper proposes a skeleton model of an alarm system where sensors have no spatial uncertainty in detecting attacks on single targets. The authors analyse how sensory information can improve the effectiveness of patrolling strategies in adversarial settings. They show that, when sensors are not affected by false negatives and false positives, the best strategy prescribes that the patroller just responds to an alarm signal rushing to the target under attack without patrolling the environment. As a consequence, in such cases the model treatment becomes trivial. On the other hand, when sensors are affected only by false negatives, the treatment can be carried out by means of an easy variation of the algorithm for the case without sensors [9]. In the last case, where false positives are admitted, the problem becomes computationally intractable. To the best of our knowledge, no previous result dealing with spatial uncertain alarm signals in adversarial patrolling is present in the literature.

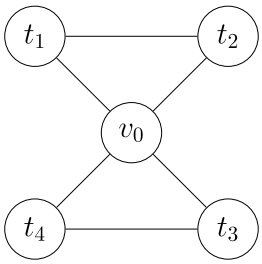
Effectively exploiting an alarm system and determining a good deployment for it (e.g., selecting the location where to install sensors) are complementary but radically different problems. The results we provide in this work lie in the scope of the first one while the treatment of the second one is left for future works. In other words, we assume that a deployed alarm system is given and we deal with the problem of strategically exploiting it at best. Any approach to search for the optimal deployment should know, in principle, how to evaluate possible deployments. In such sense, our problem needs to be addressed before one might deal with the deployment one.

1.3. Contributions

In this paper, we propose the first Security Game that integrates a spatially uncertain alarm system in game-theoretic settings for patrolling.[2] An alarm signal carries the information about the set of targets that can be under attack and it is described by the probability of being generated when each target is attacked. The analysis and the main results we propose in this work are devoted to the basic game model where the Defender can control a single patroller and the alarm system is immune to false positives and false negatives, making spatial uncertainty its only limitation. As our results show, such assumptions do not play down the significance of the arising computational challenges whose resolution is a prerequisite for more complex settings. Indeed, extensions of this model that generalize to multi-resource settings [27] and that consider false negatives [28] are built on the basic result provided in this work. The game we consider can be decomposed in a finite number of finite-horizon subgames, each called Signal Response Game from v (SRG-v) and capturing the situation in which the Defender is in a vertex v and the Attacker attacked a target, and an infinite-horizon game, called Patrolling Game (PG), in which the Defender moves in absence of any alarm signal. We show that SRG-v is FNP-hard for tree-based topologies and that, for arbitrary graphs, is APX-hard. We provide two exact algorithms. The first one, based on dynamic programming, performs a breadth-first search, while the second one, based on branch-and-bound approach, performs a depth-first search. We use the same two approaches to design two approximation algorithms.

Then, we study the PG, showing that 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. We experimentally evaluate the scalability of our exact algorithms and we compare them with the approximation ones in terms of solution quality and compute times, investigating in hard instances the gap between our hardness results and the theoretical guarantees of our approximation algorithms. We show that our approximation algorithms provide very high quality solutions even in hard instances. Finally, we provide an example of resolution for a realistic instance, based on Expo 2015, and we show that our exact algorithms can be applied for such kind of settings. Moreover, in our realistic instance we assess how the optimal patrolling strategy coincides with a static placement even when allowing a false negative rate of less or equal to 30%.

[2] A preliminary short version of this work is available in [26].



t	$\pi(t)$	$d(t)$
t_1	0.5	4
t_2	0.5	4
t_3	0.5	4
t_4	0.5	4

Fig. 1. Example of patrolling setting.

1.4. Paper structure

In Section 2, we introduce our game model. In Section 3, we study the problem of finding the strategy of the Defender for responding to an alarm signal. In Section 4, we study the patrolling problem. In Section 5, we experimentally evaluate our algorithms. In Section 6, we briefly discuss the main Security Games research directions that have been explored in the last decades. Finally, Section 7 concludes the paper. Appendix A includes the most technical proofs of the paper while Appendix B discusses some particular results obtained for special classes of instances. Appendix C reports some additional experimental results and Appendix D provides a table summarizing the notation symbols used in the paper.

2. Problem statement

In this section we formalize the problem we study. More precisely, in Section 2.1 we describe the patrolling setting and the game model, while in Section 2.2 we state the computational questions we shall address in this work.

2.1. Game model

Initially, in Section 2.1.1, we introduce a basic patrolling security game model integrating the main features from models currently studied in literature. Next, in Section 2.1.2, we extend our game model by introducing alarm signals. In Section 2.1.3, we depict the game tree of our patrolling security game with alarm signals and we decompose it in notable subgames to facilitate its study. To ease presentation, we summarized our notation symbols in Table D.3.

2.1.1. Basic patrolling security game

As is customary in the Artificial Intelligence literature [9,14], we deal with discrete, both in terms of space and time, patrolling settings, representing an approximation of a continuous environment. Specifically, we model the environment to be patrolled as an undirected connected graph $G = (V, E)$, where vertices represent different areas connected by various corridors/roads represented through the edges. Time is discretized in turns. Edges are assigned weights representing the number of time steps needed to traverse them. If not stated otherwise, we shall assume unitary weights. With $\omega^*_{i,j}$ we shall denote the shortest time to go from i to j. We define $T \subseteq V$ the subset of sensible vertices, called *targets*, that must be protected from possible attacks. Each target $t \in T$ is characterized by a value $\pi(t) \in (0, 1]$ and a penetration time $d(t) \in \mathbb{N}^+$, which measures the number of turns needed to complete an attack to t.

Example 1. We report in Fig. 1 an example of patrolling setting. Here, $V = \{v_0, v_1, v_2, v_3, v_4\}$, $T = \{t_1, t_2, t_3, t_4\}$ where $t_i = v_i$ for $i \in \{1, 2, 3, 4\}$. All the targets t present the same value $\pi(t)$ and the same penetration time $d(t)$.

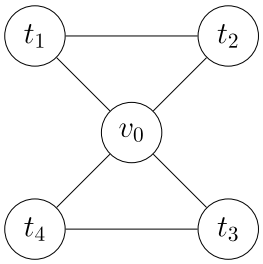
At each turn, an *Attacker* $\mathcal{A}$ and a *Defender* $\mathcal{D}$ play simultaneously having the following available actions:

- if $\mathcal{A}$ has not attacked in the previous turns, it can observe the position of $\mathcal{D}$ in the graph G[3] and decide whether to attack a target or to wait for a turn. The attack is instantaneous, meaning that there is no delay between the decision to attack and the actual presence of a threat in the selected target[4];
- $\mathcal{D}$ has no information about the actions undertaken by $\mathcal{A}$ in previous turns and selects the next vertex to patrol among those adjacent to the current one; each movement is a non-preemptive traversal of a single edge $(v, v') \in E$.

The game may conclude in correspondence of any of the two following events. The first one is when $\mathcal{D}$ patrols a target t that is under attack by $\mathcal{A}$ from less than $d(t)$ turns. In such case the attack is prevented and $\mathcal{A}$ is captured. The second

[3] Partial observability of $\mathcal{A}$ over the position of $\mathcal{D}$ can be introduced, as discussed in [29].

[4] This is a worst-case assumption according to which $\mathcal{A}$ is as strong as possible. It can be relaxed by associating execution costs to the $\mathcal{A}$'s actions, as shown in [30].



t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$
t_1	0.5	4	1
t_2	0.5	4	1
t_3	0.5	4	1
t_4	0.5	4	1

(a) Alarm system with a single signal for all the targets.

t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$	$p(s_2 \mid t)$	$p(s_3 \mid t)$	$p(s_4 \mid t)$	$p(s_5 \mid t)$
t_1	0.5	4	0.1	0.6	0.1	0.1	0.1
t_2	0.5	4	0.1	0.1	0.6	0.1	0.1
t_3	0.5	4	0.1	0.1	0.1	0.6	0.1
t_4	0.5	4	0.1	0.1	0.1	0.1	0.6

(b) Alarm system with multiple signals.

Fig. 2. Examples of alarm systems.

one is when target t is attacked and $\mathcal{D}$ does not patrol t during the $d(t)$ turns that follow the beginning of the attack. In such case, the attack is successful and $\mathcal{A}$ escapes without being captured. When $\mathcal{A}$ is captured, $\mathcal{D}$ receives a utility of 1 and $\mathcal{A}$ receives a utility of 0. When an attack to t has success, $\mathcal{D}$ receives $1 - \pi(t)$ and $\mathcal{A}$ receives $\pi(t)$. The game may not conclude if $\mathcal{A}$ decides to wait for every observed position of $\mathcal{D}$, thus never attacking. In such case, $\mathcal{D}$ receives 1 and $\mathcal{A}$ receives 0. (Another intuitive way to think at this payoff structure is to assume that $\mathcal{D}$ receives an initial utility of 1 and then loses $\pi(t)$ whenever target t is successfully compromised.) Notice that the game is constant sum and therefore it is equivalent to a zero-sum game through a positive affine transformation.

The above game model is in extensive form (being played sequentially), with imperfect information ($\mathcal{D}$ not observing the actions undertaken by $\mathcal{A}$), and with infinite horizon ($\mathcal{A}$ being in the position to wait forever). The fact that $\mathcal{A}$ can observe the actions undertaken by $\mathcal{D}$ before acting makes the *leader–follower* equilibrium the natural solution concept for our problem, where $\mathcal{D}$ is the *leader* and $\mathcal{A}$ is the *follower*. Since we focus on zero-sum games, the leader's strategy at the leader–follower equilibrium is its maxmin strategy and it can be found by employing linear mathematical programming, which requires polynomial time in the number of actions available to the players [31].

2.1.2. Introducing alarm signals

We extend the game model described in the previous section by introducing a *spatial uncertain alarm system* that can be exploited by $\mathcal{D}$. The basic idea is that an alarm system uses a number of sensors spread over the environment to gather information about possible attacks and raises an alarm signal at any time an attack occurs. The alarm signal provides some information about the location (target) where the attack is ongoing, but it is affected by uncertainty. In other words, the alarm system detects an attack but it is uncertain about the target under attack. Formally, the alarm system is defined as a pair (S, p), where $S = \{s_1, \cdots, s_m\}$ is a set of $m \geq 1$ *signals* and $p : S \times T \to [0, 1]$ is a function that specifies the probability of having the system generating a signal s given that target t has been attacked, we denote such probability with $p(s \mid t)$. 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 assume that:

- the alarm system is not affected by false positives (signals generated when no attack has occurred). Formally, $p(s \mid \triangle) = 0$, where $\triangle$ indicates that no targets are under attack;
- the alarm system is not affected by false negatives (signals not generated even though an attack has occurred). Formally, $p(\perp \mid t) = 0$, where $\perp$ indicates that no signals have been generated; in Section 4 we will show that the optimal strategies we compute under this assumption can preserve optimality even in presence of non-negligible false negatives rates.

Example 2. We report two examples of alarm systems for the patrolling setting depicted in Fig. 1. The first example is reported in Fig. 2(a). It is a low-accuracy alarm system that generates the same signal anytime a target is under attack and therefore that does not provide any information about the target under attack. The second example is reported in Fig. 2(b). It provides more accurate information about the localization of the attack than the previous example. Here, the reception of a signal s_i implies, under a uniform strategy of $\mathcal{A}$, a high probability of an attack on target t_i. Namely, when t_i is attacked the alarm system generates s_i with high probability and a different signal otherwise.

Given the presence of an alarm system defined as above, the game mechanism changes in the following way. At each turn, before deciding its next move, $\mathcal{D}$ observes whether or not a signal has been generated by the alarm system and then makes its decision considering such information. This introduces in our game a node of chance implementing the non-deterministic selection of signals.

2.1.3. The game tree and its decomposition

Here we depict the game tree of our game model, decomposing it in some recurrent subgames. A portion of the game is depicted in Fig. 3. Such tree can be read along the following steps.

- *Root of the tree.* $\mathcal{A}$ decides whether to wait for a turn (this action is denoted as $\triangle$ to indicate that no target is attacked) or to attack a target $t \in T$ (this action is denoted by the label t of the attacked target).
- *Second level of the tree.* $\mathcal{N}$ denotes the alarm system, represented by a nature-type player. Its behavior is *a priori* specified by the conditional probability mass function p, which determines the generated signal given the attack performed by $\mathcal{A}$. In particular, it is useful to distinguish between two cases:
  (a) if $\mathcal{A}$ does not attack (action $\triangle$), then no signal will be generated under the assumption that $p(\perp \mid \triangle) = 1$;
  (b) if $\mathcal{A}$ attacks target t, a signal s will be drawn from $S(t)$ with probability $p(s \mid t)$ (recall that $p(\perp \mid t) = 0$).
- *Third level of the tree.* $\mathcal{D}$ observes the signal generated by the alarm system and decides the next vertex to patrol among those adjacent to the current one (the current vertex is initially chosen by $\mathcal{D}$).
- *Fourth level of the tree and on.* It is useful to distinguish between two cases:
  (a) if no attack is present, then the tree of the subgame starting from here is the same of the tree of the whole game, except for the position of $\mathcal{D}$ that may be different from the initial one (notice that in this case each of $\mathcal{A}$'s decision nodes is a singleton information set, modeling the fact that $\mathcal{A}$ can observe $\mathcal{D}$'s position);
  (b) if an attack is taking place on target t, then only $\mathcal{D}$ can act.

Such game tree can be decomposed in a number of finite recurrent subgames such that the best strategies of the agents in each subgame are independent from those in other subgames. This decomposition allows us to apply a *divide et impera* approach, simplifying the resolution of the problem of finding an equilibrium. More precisely, we denote with Γ one of these subgames. We define Γ as a game subtree that can be extracted from the tree of Fig. 3 as follows. Given $\mathcal{D}$'s current vertex $v \in V$, select a decision node for $\mathcal{A}$ and call it i. Then, extract the subtree rooted in i discarding the branch corresponding to action Δ (no attack).[5] Intuitively, such subgame models the players' interaction when the Defender is in some given vertex v and the Attacker will perform an attack. As a consequence, each Γ obtained in such way is finite (once an attack on t started, the maximum length of the game is $d(t)$). Moreover, the set of *different* Γs we can extract is finite since we have one subgame for each possible current vertex for $\mathcal{D}$. As a consequence, we can extract at most $|V|$ different subgames. Notice that, due to the infinite horizon, each subgame can recur an infinite number of times along the game tree. However, being such repetitions independent and the game zero-sum, we only need to solve one subgame to obtain the optimal strategy to be applied in each of its repetitions. In other words, when assuming that an attack will be performed, the agents' strategies can be split in a number of independent strategies solely depending on the current position of $\mathcal{D}$. The reason why we discarded the branch corresponding to action Δ in each subgame is that we seek to deal with such complementary case exploiting a simple backward induction approach, as explained later.

First, we call *Signal Response Game* from v the subgame Γ defined as above and characterized by a vertex v representing the current vertex of $\mathcal{D}$ (for brevity, we shall use SRG-v). In an SRG-v, the goal of $\mathcal{D}$ is to find the best strategy starting from vertex v to respond to any alarm signal. All the SRG-vs are independent and thus the best strategy in each subgame does not depend on the strategies of the other subgames. The intuition is that the best strategy in an SRG-v does not depend on the vertices visited by $\mathcal{D}$ before the attack. Given an SRG-v, we denote by $\sigma^{\mathcal{D}}_{v,s}$ the strategy of $\mathcal{D}$ once observed signal s, by $\sigma^{\mathcal{D}}_{v}$ the strategy profile $\sigma^{\mathcal{D}}_{v} = (\sigma^{\mathcal{D}}_{v,s_1} \dots, \sigma^{\mathcal{D}}_{v,s_m})$ of $\mathcal{D}$, and by $\sigma^{\mathcal{A}}_{v}$ the strategy of $\mathcal{A}$. Let us notice that in an SRG-v, given a signal s, $\mathcal{D}$ is the only agent that plays and therefore each sequence of moves can be collapsed in a single action. Thus, SRG-v is essentially a two-level game in which $\mathcal{A}$ decides the target to attack and $\mathcal{D}$ decides the sequence of moves on the graph.

Then, according to classical backward induction arguments [32], once we have found the best strategies of each SRG-v, we can substitute the subgames with the agents' equilibrium utilities and then we can find the best strategy of $\mathcal{D}$ for patrolling the vertices whenever no alarm signal has been raised and the best strategy of attack for $\mathcal{A}$. We call this last problem *Patrolling Game* (for conciseness, we shall use PG). We denote by $\sigma^{\mathcal{D}}$ and $\sigma^{\mathcal{A}}$ the strategies of $\mathcal{D}$ and $\mathcal{A}$, respectively, in the PG.

[5] Rigorously speaking, our definition of subgame is not compliant with the definition provided in Game Theory [32], which requires that all the actions of a node belong to the same subgame (and therefore we could not separate action Δ from actions t). However, we can slightly change the structure of our game making our definition of subgame compliant with the one from game theory. More precisely, it is sufficient to split each node of $\mathcal{A}$ into two nodes: in the first $\mathcal{A}$ decides to attack a target or to wait for one turn, and in the second, conditioned to the fact that $\mathcal{A}$ decided to attack, $\mathcal{A}$ decides which target to attack. This way, the subtree whose root is the second node of $\mathcal{A}$ is a subgame compliant with game theory. It can be easily observed that this change to the game tree structure does not affect the behavior of the agents.

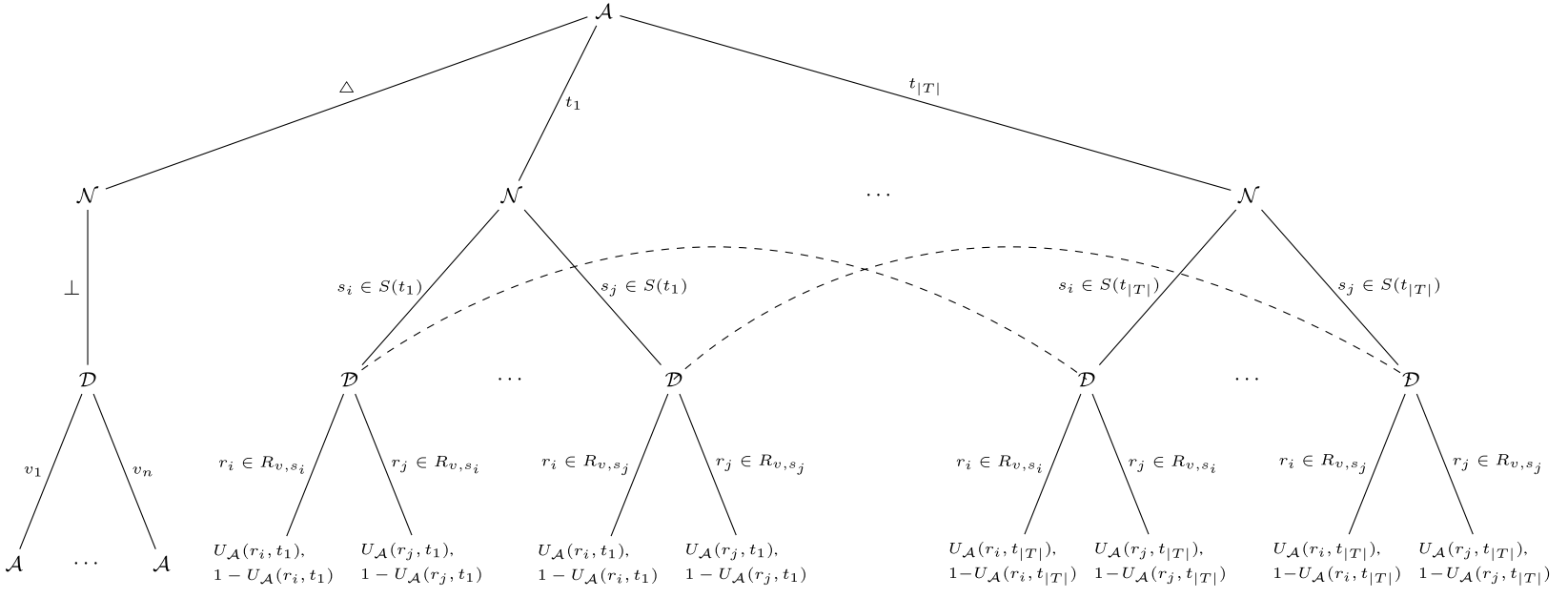


Fig. 3. Game tree, v is assumed to the be current vertex for $\mathcal{D}$, r is a collapsed sequence of moves, called *route*, that $\mathcal{D}$ performs on the graph, dashed lines indicate information sets, and function $U_{\mathcal{A}}(r_i, t_j)$ returns $\mathcal{A}$'s payoff when $\mathcal{A}$ attacks target t_j and $\mathcal{D}$ patrols on route r_i.

2.2. The computational questions we pose

Our contributions focus on the design of algorithms to find an equilibrium for our game and develop along four central questions. The first one stems directly from our problem definition.

Question 1. *Which is the best patrolling strategy for $\mathcal{D}$ maximizing its expected utility?*

Clearly, this problem is related to what we called PG in our game decomposition. In order to build an answer, we pose other three questions that, instead, involve the other subgame called SRG-v.

Question 2. *Given a starting vertex v and a signal s, is there any strategy allowing $\mathcal{D}$ to visit all the targets in $T(s)$, each within its deadline?*

Question 3. *Given a starting vertex v and a signal s, is there any pure strategy giving $\mathcal{D}$ an expected utility of at least k?*

Question 4. *Given a starting vertex v and a signal s, is there any mixed strategy giving $\mathcal{D}$ an expected utility of at least k?*

These questions are not independent from each other. In particular, answering to the last three is a prerequisite to solving the first one. For this reason, we shall take a bottom-up approach answering the above questions starting from the last three and then dealing with the first one at the whole-game level. Also, Questions 3 and 4 are not easier than Question 2 so hardness results for this last one can be extended to the others.

3. Signal response game

In this section we start by dealing with the resolution of SRG-v. Specifically, in Section 3.1 we prove the hardness of the problem, analyzing its computational complexity. Then, in Section 3.2 and in Section 3.3 we propose two algorithms, the first based on *dynamic programming* (breadth-first search) while the second adopts a *branch and bound* (depth-first search) paradigm. Furthermore, we provide a variation for each algorithm, approximating the optimal solution. For the sake of presentation, the most technical proofs are reported in Appendix A.

3.1. Complexity results

The aim of this section is to assess the computational complexity of finding an exact or approximate equilibrium for our game model. Furthermore, we aim at identifying the source of hardness of our problem and the most efficient algorithm for solving it.

Each SRG-v is characterized by $|T|$ actions available to $\mathcal{A}$, each corresponding to a target t, and by $O(|V|^{\max_t\{d(t)\}})$ decision nodes of $\mathcal{D}$. The portion of game tree played by $\mathcal{D}$ can be safely reduced by observing that $\mathcal{D}$ will move between any two targets along a shortest path. This allows us to discard from the tree all the decision nodes where $\mathcal{D}$ occupies a non-target vertex. Nevertheless, this reduction keeps the size of the game tree exponential in the parameters of the game, specifically $O(|T|^{|T|})$.[6] Notice that, the exponential size of the game tree does not constitute a proof that finding the equilibrium strategies of an SRG-v requires exponential time in the parameters of the game because it does not exclude the existence of some compact representation of $\mathcal{D}$'s strategies, e.g., Markovian strategies. The first result we provide implies that such a representation is very unlikely to exist or that, if it exists, it unlikely can be computed in polynomial time. Call g_v the expected utility of $\mathcal{A}$ from SRG-v ($1 - g_v$ is then the corresponding utility for $\mathcal{D}$). We define the following problem.

Definition 1 *(k–SRG-v).* The decision problem k–SRG-v is defined as:
INSTANCE: an instance of SRG-v;
QUESTION: is there any $\sigma^{\mathcal{D}}$ such that, when $\mathcal{A}$ plays its best response, it holds that $g_v \leq k$?

Theorem 1. *k–SRG-v is* NP*-hard even when $|S| = 1$ and the graph is a tree.*[7]

The above result shows that w.r.t. tree graphs:

- answering to Question 1 is FNP-hard,
- answering to Questions 2, 3, 4 is NP-hard.

[6] A more accurate bound is $O(|T|^{\min\{|T|,\max_t\{d(t)\}\}})$.

[7] Rigorously speaking, we show NP-hardness for a more specific class of trees we call S2L–STARS. Details can be found in A.1.

For the sake of completeness, notice that a maxmin strategy with support upper bounded by $|T|$ (that is, the number of actions available to the min player $\mathcal{A}$) always exists [33].

Now, given that we established that the main source of hardness of the problem is computing the strategy space of $\mathcal{D}$, we focus on the problem of finding an efficient representation for it. We start with some definitions.

Definition 2 *(Route).* Given a starting vertex v and a signal s, a route r is a finite sequence of vertices where, called $r(i)$ the i-th vertex, $r(0) = v$ and $r(i) \in T(s)$ for any $i > 0$. With a slight overload of notation, call $T(r)$ the set of targets from the sequence.

We restrict our attention to a special class of routes that we call *covering*. For a route r and $i > 0$, define $A_r(r(i)) = \sum_{h=0}^{i-1} \omega^*_{r(h),r(h+1)}$. Such value gives the time needed by $\mathcal{D}$ to arrive at target $r(i)$ after following a graph walk along the shortest paths between consecutive vertices of the sequence $r(0), r(1), \ldots, r(i)$.

Definition 3 *(Covering route).* A covering route is a route r where $A_r(t) \leq d(t)$ for any $t \in T(r)$.

Covering routes are abstractions for $\mathcal{D}$'s available pure strategies when the current vertex is v and a signal s has been generated. If r is a covering route for vertex v and signal s and $T(r) \subseteq T(s)$ is the set of targets in r, then we can always instantiate r to a graph walk for $\mathcal{D}$ (a sequence of moves on G) that guarantees to capture $\mathcal{A}$ on any target in $T(r)$. Such walk is simply obtained by starting from $r(0) = v$ and then covering any shortest path between $r(i)$ and $r(i+1)$. The total temporal cost of such walk is expressed by $c(r) = A_r(r(|T(r)|))$. We shall informally refer to such process as *walking a route* r.

Covering routes then constitute the action space for $\mathcal{D}$ is a SRG-v game. Even when considering a single signal s, the number such of such routes is $O(|T(s)|^{|T(s)|})$ in the worst case. However, some covering routes will never be played by $\mathcal{D}$ due to any of the following two dominance arguments [34] and discarding dominated routes is crucial in the design of an efficient algorithm.

Definition 4 *(Intra-set dominance).* Given a starting vertex v, a signal s and two different covering routes r, r' such that $T(r) = T(r')$, if $c(r) \leq c(r')$ then r dominates r'.

Definition 5 *(Inter-set dominance).* Given a starting vertex v, a signal s and two different covering routes r, r', if $T(r) \supset T(r')$ then r dominates r'.

Furthermore, it is convenient to introduce the concept of *covering set*, which is strictly related to the concept of covering route. It is defined as follows.

Definition 6 *(Covering set).* Given a starting vertex v and a signal s, a covering set (covset) Q is a subset of $T(s)$ such that there exists a covering route r with $T(r) = Q$.

Let us focus on Definition 4. It suggests that we can safely use only one route per covering set. Covering sets suffice for describing all the outcomes of the game, since the agents' payoffs depend only on the fact that $\mathcal{A}$ attacks a target t that is covered by $\mathcal{D}$ or not, 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 restricting on covering sets should be able to determine whether or not a set of targets is a covering one, which is a difficult problem as well.

Definition 7 *(COV-SET).* The decision problem COV-SET is defined as:
INSTANCE: an instance of SRG-v with a target set T;
QUESTION: is T a covering set? (Equivalently, does T admit any covering route r?)

Theorem 2. *COV-SET is* NP*-complete.*

Therefore, computing a covering route for a given set of targets (or deciding that no covering route exists) is not doable in polynomial time unless $\mathsf{P} = \mathsf{NP}$. This shows that, while covering sets suffice for defining the payoffs of the game and therefore the size of the payoffs matrix can be bounded by the number of covering sets, in practice we also need covering routes to certify that a given subset of targets is covering. The impossibility to confine our algorithms to the space of covering sets seems to suggest a complexity worse than $O(2^{|T(s)|})$. However, in the next section we provide an algorithm with complexity $O(2^{|T(s)|})$ (neglecting polynomial terms) to enumerate all and only the covering sets and, for each of them, the associated covering route with minimum cost.

Let us focus on Definition 5. Inter-Set dominance can be leveraged to introduce the concept of *maximal* covering sets (and routes) which could enable a further reduction in the set of actions available to $\mathcal{D}$.

Definition 8 *(MAXIMALITY).* Given a covering set $Q = T(r)$ for some r, we say that Q and r are maximal if there is no other covering 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 (and we can safely restrict to a single minimum cost covering route over that set), while the number of covering sets (including the non-maximal ones) 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 composed 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 searching for covering routes for any set of targets that strictly contains Q. When designing an algorithm to solve this problem, Definition 5 could then be exploited to introduce pruning techniques to save average compute time. However, the following result shows that deciding if a covering set is maximal is hard.

Definition 9 *(MAX–COV-SET).* The decision problem MAX–COV-SET is defined as:
INSTANCE: an instance of SRG-v with a target set T and a covering set $T' \subset T$;
QUESTION: is T' maximal?

Theorem 3. *There is no polynomial-time algorithm for MAX–COV-SET unless* $\mathsf{P} = \mathsf{NP}$.

Nevertheless, we show hereafter that there exists an algorithm computing 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. Therefore, neglecting polynomial terms, our algorithm has a complexity of $O(2^{|T(s)|})$.

Finally, we focus on the complexity of approximating the best solution in an SRG-v. When $\mathcal{D}$ restricts its strategies to be pure, the problem is clearly not approximable in polynomial time even when the approximation ratio depends on $|T(s)|$. The basic intuition is that, if a game instance admits the maximal covering route that covers all the targets and the value of all the targets is 1, then either the maximal covering route is played returning a utility of 1 to $\mathcal{D}$ or any other route is played returning a utility of 0, but no polynomial-time algorithm can find the maximal covering route covering all the targets, unless $\mathsf{P} = \mathsf{NP}$. On the other hand, it is interesting to investigate the case in which no restriction to pure strategies is present. We show that the problem keeps being hard.

Theorem 4. *The optimization version of k–SRG-v, say OPT–SRG-v, is* APX*-hard even in the restricted case in which the graph is metric, there is only one signal s, all targets* $t \in T(s)$ *have the same penetration time* $d(t)$*, and there exists a maximal covering route covering all the targets.*

The above theorem allows us to make two important remarks.

Remark 1. The above result does not exclude the existence of constant-ratio approximation algorithms for OPT–SRG-v. We conjecture that it is unlikely. OPT–SRG-v presents similarities with the (metric) DEADLINE–TSP, where the goal is to find the longest path of vertices each traversed before its deadline. The DEADLINE–TSP does not admit any constant-ratio approximation algorithm [35] and the best-known approximation algorithm has logarithmic approximation ratio [36]. The following observations can be produced about the relationships between OPT–SRG-v and DEADLINE–TSP:

- when the maximal route covering all the targets in the OPT–SRG-v exists, the optimal solution of the OPT–SRG-v is also optimal for the DEADLINE–TSP applied to the same graph;
- when the maximal route covering all the targets in the OPT–SRG-v does not exist, the optimal solutions of the two problems are different, even when we restrict us to pure-strategy solutions for the OPT–SRG-v;
- approximating the optimal solution of the DEADLINE–TSP does not give a direct technique to approximate OPT–SRG-v, since we should enumerate all the subsets of targets and for each subset of targets we would need to execute the approximation of the DEADLINE–TSP, but this would require exponential time. We notice in addition that even the total number of sets of targets with logarithmic size is not polynomial, being $\Omega(2^{\log^2(|T|)})$, and therefore any algorithm enumerating them would require exponential time;
- when the optimal solution of the OPT–SRG-v is randomized, examples of optimal solutions in which maximal covering routes are not played can be produced, showing that at the optimum it is not strictly necessary to play maximal covering routes, but even approximations suffice.

Remark 2. If it were possible to map DEADLINE–TSP instances to OPT–SRG-v instances where the maximal covering route covering all the targets exists, then it would trivially follow that OPT–SRG-v does not admit any constant-approximation ratio. We were not able to find such a mapping and we conjecture that, if there is an approximation-preserving reduction

from DEADLINE–TSP to OPT–SRG-v, then we cannot restrict to such instances. The study of instances of OPT–SRG-v where mixed strategies may be optimal make the treatment very challenging.

3.2. Dynamic-programming algorithm

We start by presenting two algorithms. The first one is exact, while the second one is an approximation algorithm. Both algorithms are based on a dynamic programming approach.

3.2.1. Exact algorithm

In this section we provide an algorithm to compute the strategies available to $\mathcal{D}$ when v is the current vertex and signal s has been generated by the alarm system. The idea is to adopt a dynamic programming method that enumerates covering sets of increasing cardinalities. Each covering set is obtained by a sequence of expansions that, starting from the empty set, add one target at each iteration. We denote a covering set by $Q^k_{v,t}$ where k indicates its cardinality while v and t denote the starting vertex of $\mathcal{D}$ and the last target added to the set, respectively. The algorithm operates in such a way that the sequence of targets corresponding to the expansions made to obtain $Q^k_{v,t}$ is a covering route for that set. Informally, we shall call it the *generative route* of $Q^k_{v,t}$ and we will denote with $c(Q^k_{v,t})$ its temporal cost. We choose to obtain this by requiring any expansion to be admissible with respect to three conditions. Given a set $Q^k_{v,t}$ a new set $Q^{k+1}_{v,w} = Q^k_{v,t} \cup \{w\}$ can be obtained if:

1. w is not covered in the current covering set, that is $w \in T(s) \setminus Q^k_{v,t}$;
2. w can be covered by $d(w)$ by extending the generative route of $Q^k_{v,t}$ with a shortest walk from t to w, that is $d(w) \geq c(Q^k_{v,t}) + \omega^*_{t,w}$;
3. call t' a target visited by a shortest path from t to w, if $t' \notin Q^k_{v,t}$ then it cannot be covered, that is $d(t') < c(Q^k_{v,t}) + \omega^*_{t,t'}$.

Conditions 1 and 2 require any expansion to form a new covering set consistent with Definition 6, thus ensuring the algorithm's soundness. Condition 3 requires $Q^k_{v,t}$ to be a *proper* descriptor of its generative route, meaning that such route, once walked, covers uniquely the targets included in $Q^k_{v,t}$ and not targets outside it. This last requirement is an operational choice to reduce the number of expansions made in each iteration of the algorithm. Consider for instance a graph with degree bounded by 3, Condition 3 allows us to generate in the worst case 3 new covering sets at each expansion round instead of $|T|$. Notice that under Condition 3 the algorithm can still generate any maximal covering set.

Lemma 5. *Any maximal covering set can be generated from the empty set with a sequence of admissible expansions.*

Proof sketch. Consider an expansion which, by adding a target w, violates Condition 3 for a target t'. Such expansion can always be split in two admissible ones. The first adding t', the second adding w. The same rationale works for multiple expansions and multiple t' and clearly applies to maximal covering sets which are a subset of covering sets. □

Condition 3 can be exploited to reduce the required compute time but, rigorously speaking, it presents a drawback. To establish if target w can be added to set $Q^k_{v,t}$ the algorithm needs to check every shortest path from t to w, and such paths can be, in the worst case, exponentially many. We can cope with this by adopting the following simplification. We fix a set of canonical shortest paths by running the Floyd–Warshall algorithm. Then, given t and w, we fetch the canonical shortest path between them and we check Condition 3 assuming that such path is unique. If the condition is not verified under this assumption then it is also not verified in its original definition. If otherwise, the condition is verified, then the algorithm (assuming validity of Conditions 1 and 2) might obtain a covering set $\bar{Q}^{k+1}_{v,w}$ which is not a proper one, meaning that by walking its generative route at least one target $t' \notin \bar{Q}^{k+1}_{v,w}$ gets covered. Let us assume that this is the case and, w.l.o.g., that t' is the only invalidating target. The algorithm's current solution representation (the set $\bar{Q}^{k+1}_{v,w}$) would then be inconsistent with the actual solution (the generative route). By Lemma 5 though, we know that the algorithm would generate also set $Q^{k+2}_{v,w}$ making two admissible expansions with t' and w to $Q^k_{v,t}$. Since $Q^{k+2}_{v,w} = \bar{Q}^{k+1}_{v,w} \cup \{t'\}$ the non-proper covering set $\bar{Q}^{k+1}_{v,w}$ is removable by inter-set dominance (Definition 5). Obviously this workaround comes at an additional cost: the algorithm unnecessarily generates the set $\bar{Q}^{k+1}_{v,w}$ which under the proper definition of Condition 3 would have been avoided. Still, the above solution turned out very convenient since exponential multiplicity of shortest paths is very unlikely in graphs representing real environments.

In Algorithm 1 we provide full technical details. Covering sets obtained by the algorithm are grouped in collections: $\mathcal{C}^k_{v,t}$ denotes the collection of all covering sets of cardinality k where the last expansion added target t. After the required initialization steps (Lines 1 and 2) for any generated $Q^{k-1}_{v,t}$ we compute the set of admissible expansions Q^+ (Line 6) and we apply each one of them (Line 8). In Step 9, we make use of a procedure called $Search(Q, \mathcal{C})$ where Q is a covering set and $\mathcal{C}$ is a collection of covering sets. The procedure outputs Q if $Q \in \mathcal{C}$ and $\emptyset$ otherwise. (We adopted an efficient implementation of such procedure which can run in $O(|T(s)|)$. More precisely, we represent a covering set Q as a binary vector of length

$|T(s)|$ where the i-th component is set to 1 if target $t_i \in Q$ and 0 otherwise. A collection of covering sets C can then be represented as a binary tree with depth $|T(s)|$. The membership of a covering set Q to collection C is represented with a branch of the tree in such a way that if $t_i \in Q$ then we have a left edge at depth $i-1$ on such branch. We can easily determine if $Q \in \mathcal{C}$ by checking if traversing a left (right) edge in the tree each time we read a 1 (0) in Q's binary vector we reach a leaf node at depth $|T(s)|$. The insertion of a new covering set in the collection can be done in the same way by traversing existing edges and expanding the tree where necessary.) If the newly generated covering set is not present in its collection or is already present with a higher cost (Step 10), then collection and cost are updated (Steps 11 and 12).

Algorithm 1 DP-ComputeCovSets(v, s).

1: $\forall t \in T(s)$: $\mathcal{C}^0_{v,v} = \{\emptyset\}$
2: $c(\emptyset) = 0$
3: **for all** $k \in \{1 \dots |T(s)|\}$ **do**
4: **for all** $t \in T(s)$ **do**
5: **for all** $Q^{k-1}_{v,t} \in \mathcal{C}^{k-1}_{v,t}$ **do**
6: $Q^+ = \{w \in T(s) \mid$ Conditions 1–3 are satisfied$\}$
7: **for all** $w \in Q^+$ **do**
8: $Q^k_{v,w} = Q^{k-1}_{v,t} \cup \{w\}$
9: $U = Search(Q^k_{v,w}, \mathcal{C}^k_{v,w})$
10: **if** $c(U) > c(Q^{k-1}_{v,t}) + \omega^*_{t,w}$ **then**
11: $\mathcal{C}^k_{v,w} = \mathcal{C}^k_{v,w} \cup \{Q^k_{v,w}\}$
12: $c(Q^k_{v,w}) = c(Q^{k-1}_{v,t}) + \omega^*_{t,w}$
13: **return** $\{\mathcal{C}^k_{v,t} : t \in T(s), k \leq |T(s)|\}$

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 $Y_{|T'|} = \cup_{t \in T'}\{Search(T', \mathcal{C}^{|T'|}_{v,t})\}$ (notice that if T' is not a covering set then $c^*(T') = \infty$). For the sake of simplicity, Algorithm 1 does not specify how to carry out two sub-tasks we describe in the following.

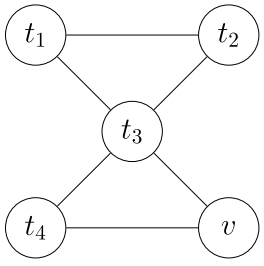
The first one is the *annotation of dominated covering sets*. Each time Steps 11 and 12 are executed, a covering set is added to some collection. Let us call it Q and assume it has cardinality k. Each time a new Q has to be included at cardinality k, we mark all the covering sets at cardinality $k-1$ that are dominated by Q (Definition 5). The number of sets that can be dominated is in the worst case $|Q|$ since each of them has to be searched in collection $\mathcal{C}^{k-1}_{v,t}$ 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, as described above, the *Search* procedure takes $O(|T(s)|)$. 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 in the next iteration.

The second task is the *generation of routes*. To do this we maintain a list of generating routes by iteratively appending terminal vertex w to the generative route of $Q^{k-1}_{v,t}$ when set $Q^{k-1}_{v,t} \cup \{w\}$ is included in its corresponding collection. At the end of the algorithm only routes that correspond to non-dominated covering sets are returned. Maintaining such a list introduces a $O(1)$ cost.

Theorem 6. *Algorithm 1 is an exact algorithm and has worst-case complexity of* $O(|T(s)|^2 2^{|T(s)|})$ *since it has to compute covering sets up to cardinality* $|T(s)|$. *With annotations of dominances and routes generation the whole algorithm yields a worst-case complexity of* $O(|T(s)|^5 2^{|T(s)|})$.

Notice that the algorithm is exact since it is based on an enumeration procedure.

Example 3. We provide a simple example of execution of Algorithm 1. Consider a problem instance with a single signal, arbitrary target values while topology and penetration times are as follow:



t	$d(t)$
t_1	3
t_2	3
t_3	3
t_4	3

We report the expansions made by the algorithm for increasing cardinalities (value of k) in the following table.

$k=0$	$k=1$	$k=2$	$k=3$
$\emptyset, c=0$	$Q^1_{v,t_3}=\{t_3\}, c=1$	$Q^2_{v,t_1}=\{t_1,t_3\}, c=2$	$Q^3_{v,t_2}=\{t_1,t_2,t_3\}, c=3$ $Q^3_{v,t_4}=\{t_1,t_3,t_4\}, c=4$
		$Q^2_{v,t_2}=\{t_2,t_3\}, c=2$	$Q^3_{v,t_1}=\{t_1,t_2,t_3\}, c=3$ $Q^3_{v,t_4}=\{t_2,t_3,t_4\}, c=4$
		$Q^2_{v,t_4}=\{t_3,t_4\}, c=2$	$Q^3_{v,t_1}=\{t_1,t_3,t_4\}, c=4, \square$ $Q^3_{v,t_2}=\{t_2,t_3,t_4\}, c=4, \lozenge$
	$Q^1_{v,t_4}=\{t_4\}, c=1$	$Q^2_{v,t_3}=\{t_3,t_4\}, c=2$	$Q^3_{v,t_1}=\{t_1,t_3,t_4\}, c=3, \blacksquare$ $Q^3_{v,t_2}=\{t_2,t_3,t_4\}, c=3, \blacklozenge$

Notice that Constraint 3 intervenes both when expanding covering sets with $k=1$ and $k=2$. In the table, c indicates the temporal cost of the relative covset while the covset marked with $\blacklozenge$ dominates the one marked with $\lozenge$ while the covset marked with $\blacksquare$ dominates the one marked with $\square$.

3.2.2. Approximation algorithm

The dynamic programming algorithm presented in the previous section cannot be directly adopted to approximate the maximal covering routes. We notice that even in the case we introduce a logarithmic upper bound over the size of the covering sets generated by Algorithm 1, we could obtain a number of routes that is $O(2^{\log^2(|T(s)|)})$, and therefore exponential. Thus, our goal is to design a polynomial-time algorithm that generates a polynomial number of *good* covering routes. We observe that if we have a total order over the vertices and we work over a complete graph of the targets where each edge corresponds to the shortest path, we can find in polynomial time the maximal covering routes subject to the constraint that, given any pair of targets t, t' in a route, t can precede t' in the route only if t precedes t' in the order. We call *monotonic* a route satisfying a given total order. Algorithm 2 returns the maximal monotonic covering routes when the total order is lexicographic (trivially, to change the order, it is sufficient to re-label the targets).

Algorithm 2 is based on dynamic programming and works as follows. $R(k,l)$ is a matrix storing in each cell one route, while $L(k,l)$ is a matrix storing in each cell the maximum lateness of the corresponding route (see below for the meaning of k and l). The maximum lateness of a route r captures the maximum delay between a target's first visit and its deadline. Formally, it is defined as $\max_{t\in T(s)} A_r(t) - d(t)$. The route stored in $R(k,l)$ is the one with the minimum lateness among all the monotonic ones covering l targets where t_k is the first visited target. Thus, basically, when $l=1$, $R(k,l)$ contains the route $\langle v, t_k\rangle$, while, when $l>1$, $R(k,l)$ is defined appending to $R(k,1)$ the best (in terms of minimizing the maximum lateness) route $R(k',l-1)$ for every $k'>k$, in order to satisfy the total order. The whole set of routes in R are returned.[8] The complexity of Algorithm 2 is $O(|T(s)|^3)$, except the time needed to find all the shortest paths.

Algorithm 2 MonotonicLongestRoute(v, s).

```
1: ∀k, k' ∈ {1, 2, ..., |T(s)|}, R(k, k') = ∅, L(k, k') = +∞, C_R(k) = ∅, C_L(k) = +∞
2: for all ∀k ∈ {|T(s)|, |T(s)| − 1, ..., 1} do
3:     for all ∀l ∈ {1, 2, ..., |T(s)|} do
4:         if l = 1 then
5:             R(k, l) = ⟨v, t_k⟩
6:             L(k, l) = ω*_{v,t_k} − d(t_k)
7:         else
8:             for all k' s.t. |T(s)| ≥ k' > |T(s)| − k do
9:                 C_R(k') = ⟨R(k, 1), R(k', l − 1)⟩
10:                C_L(k') = max{L(k, 1), ω*_{v,t_k} + ω*_{t_k,k'} − ω*_{v,k'} + L(k', l − 1)}
11:            j = arg min_j {C_L(j)}
12:            if C_L(j) ≤ 0 then
13:                R(k, l) ← C_R(j)
14:                L(k, l) ← C_L(j)
15: return R
```

We use different total orders (breaking ties randomly) over the set of targets, collecting all the routes generated using each total order:

- increasing order $\omega^*_{v,t}$: the rationale is that targets close to v will be visited before targets far from v;
- increasing order $d(t)$: the rationale is that targets with short deadlines will be visited before targets with long deadlines;
- increasing order $d(t) - \omega^*_{v,t}$ (we call this quantity *excess time*): the rationale is that targets with short excess time will be visited before targets with long excess time.

[8] We notice that dominance can be applied to discard dominated routes. However, in this case, the improvement would be negligible since the total number of routes, including the non-dominated ones, is polynomial.

Algorithm 3 Branch-and-Bound(v, s, ρ).

1: $CL_{max} \leftarrow \emptyset$
2: $CL_{min} \leftarrow \emptyset$
3: **for all** $t \in T(s)$ **do**
4: **if** $\omega^*_{v,t} \leq d(t)$ **then**
5: Tree-Search($\lceil \rho \cdot |T(s)| \rceil$, $\langle v, t \rangle$)
6: **return** CL_{max}

In addition, we use a random restart generating random permutations over the targets.

Theorem 7. *Algorithm 2 provides an approximation with ratio* $\Omega(\frac{1}{|T(s)|})$.

Proof sketch. The worst case for the approximation ratio of our algorithm occurs when the covering route including all the targets exists and each covering route returned by our heuristic algorithm visits only one target. In that case, the optimal expected utility of $\mathcal{D}$ is 1. Our algorithm, in the worst case in which $\pi(t) = 1$ for every target t, returns an approximation ratio $\Omega(\frac{1}{|T(s)|})$. It is straightforward to see that, in other cases, the approximation ratio is larger. □

3.3. Branch-and-bound algorithms

The dynamic programming algorithm presented in the previous section essentially implements a breadth-first search. In some specific situations, depth-first search could outperform breadth-first search, e.g., when penetration times are relaxed and good heuristics lead a depth-first search to find in a brief time the maximal covering route, avoiding to scan an exponential number of routes as the breadth-first search would do. In this section, we adopt the branch-and-bound approach to design both an exact algorithm and an approximation algorithm. In particular, in Section 3.3.1 we describe our exact algorithm, while in Section 3.3.2 we present the approximation one.

3.3.1. Exact algorithm

Our branch-and-bound algorithm (see Algorithm 3) is a tree-search based algorithm working on the space of the covering routes and returning a set of covering routes R. It works as follows.

Initial step. We exploit two global set variables, CL_{min} and CL_{max} initially set to empty (Steps 1–2 of Algorithm 3). These variables contain *closed* covering routes, namely covering routes which cannot be further expanded without violating the penetration time of at least one target during the visit. CL_{max} contains the covering routes returned by the algorithm, while CL_{min} is used for pruning as discussed below. Given a starting vertex v and a signal s, for each target $t \in T(s)$ such that $\omega^*_{v,t} \leq d(t)$ we generate a covering route $r = \langle v, t \rangle$ with $r(0) = v$ and $r(1) = t$ (Steps 5 of Algorithm 3). Thus, $\mathcal{D}$ has at least one covering route per target that can be covered in time from v. Notice that if, for some t, such minimal route does not exist, then target t cannot be covered (even the shortest path from the starting vertex v cannot guarantee capture). This does not guarantee that $\mathcal{A}$ will attack t with full probability since, depending on the values π, $\mathcal{A}$ could find more profitable to randomize over a different set of targets. The meaning of parameter ρ (used in Line 5 of Algorithm 3) is described below.

Route expansions. The subsequent steps essentially evolve on each branch according to a depth-first search with backtracking limited by ρ. The choice of ρ directly influences the behavior of the algorithm and consequently its complexity. Each node in the search tree represents a route r built so far starting from an initial route $\langle v, t \rangle$. At each iteration, route r is expanded by inserting a new target at a particular position. We denote with $r^+(q, p)$ the route obtained by inserting target q after the p-th target in r. Notice that every expansion of r will preserve the relative order with which targets already present in r will be visited. The collection of all the feasible expansions r^+s (i.e., the ones that are covering routes) is denoted by R^+ and it is ordered according to a heuristic that we describe below. Algorithm 6, described below, is used to generate R^+ (Step 1 of Algorithm 4). In each open branch (i.e., $R^+ \neq \emptyset$), if the depth of the node in the tree is smaller or equal to $\lceil \rho \cdot |T(s)| \rceil$ then backtracking is disabled (Steps 7–10 of Algorithm 4), while, if the depth is larger than such value, is enabled (Steps 5–6 of Algorithm 4). This is equivalent to fix the relative order of the first (at most) $\lceil \rho \cdot |T(s)| \rceil$ inserted targets in the current route. In this case, with $\rho = 0$ we do not rely on the heuristics at all, full backtracking is enabled, the tree is fully expanded and the returned R is complete, i.e., it contains all the non-dominated covering routes. Route r is repeatedly expanded in a greedy fashion until no insertion is possible. As a result, Algorithm 4 generates at most $|T(s)|$ covering routes.

Pruning. Algorithm 5 is in charge of updating CL_{min} and CL_{max} each time a route r cannot be expanded and, consequently, the associated branch must be closed. We call CL_{min} the *minimal* set of closed routes. This means that a closed route r belongs to CL_{min} only if CL_{min} does not already contain another $r' \subseteq r$. Steps 1–4 of Algorithm 5 implement such condition: first, in Steps 2–3 any route r' such that $r' \supseteq r$ is removed from CL_{min}, then route r is inserted in CL_{min}. Routes in CL_{min} are used by Algorithm 6 in Steps 2 and 6 for pruning during the search. More precisely, a route r is not expanded with a target q at position p if there exists a route $r' \in CL_{min}$ such that $r' \subseteq r^+(q, p)$. This pruning rule is safe since by definition if $r' \in CL_{min}$, then all the possible expansions of r' are unfeasible and if $r' \subseteq r$ then r can be obtained by expanding from r'. This pruning mechanism explains why once a route r is closed is always inserted in CL_{min} without checking the

Algorithm 4 Tree-Search(k, r).

```
1: R+ = {r(1), r(2), ...} ← Expand(r)
2: if R+ = ∅ then
3:     Close(r)
4: else
5:     if k > 0 then
6:         Tree-Search (k − 1, r(1))
7:     else
8:         for all r+ ∈ R+ do
9:             Tree-Search (0, r+)
10:            Close(r+)
```

Algorithm 5 Close(r).

```
1: for all r' ∈ CL_min do
2:     if r ⊆ r' then
3:         CL_min = CL_min \ {r'}
4: CL_min = CL_min ∪ {r}
5: for all r' ∈ CL_max do
6:     if r ⊆ r' then
7:         return
8: CL_max = CL_max ∪ {r}
```

insertion against the presence in CL_{min} of a route r'' such that $r'' \subseteq r$. Indeed, if such route r'' would be included in CL_{min} we would not be in the position of closing r, having r being pruned before by Algorithm 6.

We use CL_{max} to maintain a set of the generated *maximal* closed routes. This means that a closed route r is inserted here only if CL_{max} does not already contain another r' such that $r' \supseteq r$. This set keeps track of closed routes with maximum number of targets. Algorithm 5 maintains this set by inserting a closed route r in Step 12 only if no route $r' \supseteq r$ is already present in CL_{max}. Once the whole algorithm terminates, CL_{max} contains the final solution.

Heuristic function. A key component of this algorithm is the heuristic function that drives the search. The heuristic function is defined as $h_r : \{T(s) \setminus T(r)\} \times \{1 \dots |T(r)|\} \to \mathbb{Z}$, where $h_r(t', p)$ evaluates the cost of expanding r by inserting target t' after the p-th target of r. The basic idea, inspired by [37], 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. We use the two sets according to the following rules (see Algorithm 6):

- the insertion of a target belonging to T_{tight} is always preferred to the insertion of a target belonging to T_{large}, independently of the insertion position;
- insertions of $t \in T_{tight}$ are ranked according to h considering first the insertion position and then the target;
- insertions of $t \in T_{large}$ are ranked according to h considering first the target and then the insertion position.

The rationale behind this rule is that targets with a tight penetration time should be inserted first and at their best positions. On the other hand, targets with a large penetration time can be covered later. Therefore, in this last case, it is less important which target to cover than when to cover it.

Theorem 8. *Algorithm 3 with $\rho = 0$ is an exact algorithm and has an exponential computational complexity since it builds a full tree of covering routes, with worst-case size $O(|T(s)|^{|T(s)|})$.*

Algorithm 6 Expand(r).

if $T_{tight} \not\subseteq T(r)$ **then**
  for all $q \in T_{tight} \setminus T(r)$ **do**
    $P_q = \{ p_q^{(1)}, p_q^{(2)}, \dots p_q^{(b)} \}$ s.t. $\forall i \in \{1, \dots, b\}$, $\begin{cases} h_r(q, p_q^{(i)}) \geq h_r(q, p_q^{(i+1)}) \\ r^+(q, p_q^{(i)}) \text{ is a covering route} \\ \nexists v' \in CL_{min} : r' \subseteq r^+(q, p_q^{(i)}) \end{cases}$
    $Q = \{q^{(1)}, q^{(2)}, \dots, q^{(c)}\}$ s.t. $\forall i \in \{1, \dots, c\}$, $h_r(q^{(i)}, p_{q^{(i)}}^{(1)}) \geq h_r(q^{(i+1)}, p_{q^{(i+1)}}^{(1)})$
    $R^+ = \{r^{(1)}, r^{(2)}, \dots r^{(k)}\}$ where $\begin{cases} r^{(1)} &= r^+(q^{(1)}, p_{q^{(1)}}^{(1)}) \\ \cdots &= \cdots \\ r^{(k)} &= r^+(q^{(c)}, p_{q^{(c)}}^{(b)}) \end{cases}$
if $T_{large} \not\subseteq T(r)$ **then**
  for all $u \in T_{large} \setminus T(r)$ **do**
    $Q_p = \{ q_p^{(1)}, q_p^{(2)}, \dots q_p^{(b)} \}$ s.t. $\forall i \in \{1, \dots, b\}$, $\begin{cases} h_r(q_p^{(i)}, p) \geq h_r(q_p^{(i+1)}, p) \\ r^+(q_p^{(i)}, p) \text{ is a covering route} \\ \nexists r' \in CL_{min} : r' \subseteq r^+(q_p^{(i)}, p) \end{cases}$
    $P = \{p^{(1)}, p^{(2)}, \dots, p^{(c)}\}$ s.t. $\forall i \in \{1, \dots, c\}$, $h_r(q^{(1)}, p_{q^{(1)}}^{(i)}) \geq h_r(q^{(1)}, p_{q^{(1)}}^{(i+1)})$
    $R^+ = R^+ \cup \{r^{(k+1)}, r^{(k+2)}, \dots r^{(K)}\}$ where $\begin{cases} r^{(k+1)} &= r^+(q_p^{(1)}, p^{(1)}) \\ \cdots &= \cdots \\ r^{(K)} &= r^+(q_p^{(b)}, p^{(c)}) \end{cases}$
return R^+

3.3.2. Approximation algorithm

Since ρ determines the *completeness degree* of the generated tree, we can exploit Algorithm 3 tuning ρ to obtain an approximation algorithm that is faster w.r.t. the exact one.

In fact, when $\rho < 1$ completeness is not guaranteed in favor of a less computational effort. In this case, the only guarantees that can be provided for each covering route $r \in CL_{max}$, once the algorithm terminates are:

- no other $r' \in CL_{max}$ dominates r;
- no other $r' \notin CL_{max}$ such that $r \subseteq r'$ dominates r. Notice this does not prevent the existence of a route r'' not returned by the algorithm that visits targets $T(r)$ in a different order and that dominates r.

When ρ is chosen as $\frac{k}{|T(s)|}$ (where $k \in \mathbb{N}$ is a parameter), the complexity of generating covering routes becomes polynomial in the size of the input. We can state the following theorem, whose proof is analogous to that one of Theorem 7.

Theorem 9. *Algorithm 4 with $\rho = \frac{k}{|T(s)|}$ provides an approximation with ratio $\Omega(\frac{1}{|T(s)|})$ and runs in $O(|T(s)|^3)$ given that heuristic h_r can be computed in $O(|T(s)|^2)$.*

3.4. Solving SRG-v

Now we can formulate the problem of computing the optimal signal–response strategy for $\mathcal{D}$. Let us denote with $\sigma_{v,s}^{\mathcal{D}}(r)$ 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 $U_{\mathcal{A}}(r, t)$, representing the utility function of $\mathcal{A}$ and defined as follows:

$$U_{\mathcal{A}}(r, t) = \begin{cases} \pi(t) & \text{if } t \notin r \\ 0 & \text{otherwise} \end{cases}.$$

The best $\mathcal{D}$'s strategy (i.e., the maxmin strategy) can be found by solving the following linear mathematical programming problem:

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

The size of the mathematical program is composed of $|T| + |S|$ constraints (excluded ≥ 0 constraints) and $O(|V||S| \max_{v,s}\{|R_{v,s}|\})$ variables. This shows that the hardness is due only to $\max_{v,s}\{|R_{v,s}|\}$, which, in its turn, depends only on $|T(s)|$. We provide the following remark.

Remark 3. We observe that the discretization of the environment as a graph is as accurate as the number of vertices is large, corresponding to reduce the size of the areas associated with each vertex, as well as to reduce the temporal interval

associated with each turn of the game. Our algorithms show that increasing the accuracy of the model in terms of number of vertices requires polynomial time.

4. Patrolling game

In this section, we focus on the PG. Specifically, in Section 4.1 we state our main result showing that patrolling is not necessary when an alarm system is present, in Section 4.2 we propose the algorithm to deal with the PG, in Section 4.3 we summarize the complexity results about Questions 1–4.

4.1. Stand still

We focus on the problem of finding the best patrolling strategy given that we know 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 [9]. We define $v^* = \arg\min_{v \in V}\{g_v\}$, where g_v is the value returned by the optimization problem described in Section 3.4, 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 10. *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 that, in turn, will raise a response yielding an 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 a signal is the best strategy for $\mathcal{D}$. □

The rationale is based upon the fact that, without false positives and missed detections, the signal response strategy solely depends on the current vertex occupied by $\mathcal{D}$ and on the generated signal (not depending then on previously visited vertices). So, if the patrolling strategy of $\mathcal{D}$ prescribes to patrol a set of vertices, say V', then, since $\mathcal{A}$ can observe the position of $\mathcal{D}$, the best strategy of $\mathcal{A}$ is to wait for $\mathcal{D}$ being in $v' = \arg\max_{v \in V'}\{g_v\}$ and then to attack. Thus, by definition of g_{v^*}, if $\mathcal{D}$ leaves v^* to patrol additional vertices the expected utility it receives is no larger than that it receives from staying in v^*.

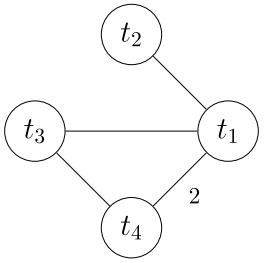
The validity of Theorem 10 goes beyond the model studied in this work. Indeed, it can be easily shown how such results keep valid when having multiple resource controlled by $\mathcal{D}$. Also, a deeper analysis of Theorem 10 can show that its scope does include cases where missed detections are present 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 $\alpha = 0.5$.*

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 gives 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:

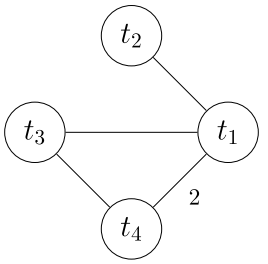
$$(1-\alpha)(1-g_{v^*}) > (1-g_{v_2^*}). \tag{1}$$

To prove Proposition 1, it then suffices to provide a Patrolling Game instance where Equation (1) holds under some non-null missed detection rate α. In Fig. 4(a) and Fig. 4(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. 4(a) it holds that $g_{v^*} = 0$ and $g_{v_2^*} = 0.75$, where $v^* = 4$ and $v_2^* = 1$. Thus, by Equation (1), placement $v^* = 4$ is the optimal strategy for $\alpha \leq 0.25$. Under the same reasoning scheme, in Fig. 4(b) we have that $g_{v^*} = 0$ and $g_{v_2^*} = 0.5$, making the placement $v^* = 4$ optimal for any $\alpha \leq 0.5$. □



t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$
t_1	0.5	1	1.0
t_2	0.5	3	1.0
t_3	0.5	2	1.0
t_4	0.5	2	1.0

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



t	$\pi(t)$	$d(t)$	$p(s_1 \mid t)$
t_1	1.0	1	1.0
t_2	1.0	3	1.0
t_3	1.0	2	1.0
t_4	1.0	2	1.0

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

Fig. 4. Two examples proving Proposition 1.

Algorithm 7 BestPlacement(G, s).

```
1: U(v) ← 0 for every v ∈ V
2: for all v ∈ V do
3:     U(v) ← SolveSRG(v)
4: return max(U)
```

It is reasonable to expect that a similar result holds also for the case with false positives. However, dealing with false positives is much more intricate than handling false negatives and requires new models. For example, $\mathcal{D}$ could respond to an alarm signal only with a given probability and with the remaining probability could stay in the current vertex. For this reason, we leave the treatment of false positives and a more accurate treatment of false negatives to future works.

4.2. Computing the best placement

Under the absence of false positives and missed detections, Theorem 10 simplifies the computation of the patrolling strategy by reducing it to the problem of finding v^*. To such aim, we must solve a SRG-v for each possible starting vertex v and select the one with the maximum expected utility for $\mathcal{D}$. Algorithm 7 depicts the solving algorithm. Function $SolveSRG(v)$ returns the optimal value $1 - g_{v^*}$. The complexity is linear in $|V|$, once g_v has been calculated for every v.

Since all the vertices are possible starting points, we should face this hard 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 starting vertex v could help us finding the solution for another starting vertex v'. In other words, 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'. This would allow us to solve an exponential-time problem only once instead of solving it for each vertex of the graph. To answer this question, we resort to hardness results for reoptimization, also called *locally modified* problems [38]. We show that, even if we know all the covering routes from a starting vertex, once we changed the starting vertex selecting an adjacent one, finding the covering routes from the new starting vertex is hard.

Definition 10 *(LM–COV–ROUTE).* A locally modified covering route (LM–COV–ROUTE) problem is defined as follows:
INSTANCE: graph $G = (V, E)$, a set of targets T with penetration times d, two starting vertices v_1 and v_2 that are adjacent, and a covering route r_1 with $r_1(0) = v_1$ such that $T(r_1) = T$.
QUESTION: is there r_2 with $r_2(0) = v_2$ and $T(r_2) = T$?

Theorem 11. LM–*COV–ROUTE is* NP-*complete.*

This shows that iteratively applying Algorithm 1 to SRG-v for each starting vertex v and then choosing the vertex with the highest utility is the best we can do in the worst case.

Table 1
Computational complexity of discussed questions.

Question	Topology	
	Tree	Arbitrary
Question 1	FNP-hard	APX-hard
Question 2	NP-hard	NP-hard
Question 3	NP-hard	NP-hard
Question 4	NP-hard	NP-hard
Question 2	LM?	NP-hard

4.3. Summary of results

We summarize our computational results about Questions 1–4 in Table 1, including also results about the resolution of the PG. We use '?' for the problems remained open in this paper.

5. Experimental evaluation

We implemented our algorithms in MATLAB and we used a 2.33 GHz LINUX machine to run our experiments. For a better analysis, we provide two different experimental evaluations. In Section 5.1, we apply our algorithms to worst-case instances, in order to evaluate the worst-case performance of the algorithms and to investigate experimentally the gap between our APX-hardness result and the theoretical guarantees of our approximation algorithms. In Section 5.2, we apply our algorithms to a specific realistic instance we mentioned in Section 1, Expo 2015.

5.1. Worst-case instances analysis

We evaluate the scalability of Algorithm 1 and the quality of the solution returned by our approximation algorithms 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. In the next section we describe our experimental setting, in Section 5.1.2 we provide a quantitative analysis of the exact algorithms while in Section 5.1.3 we evaluate the quality of our approximations.

5.1.1. Setting

We can build hard instances for our problem from instances of HAMILTONIAN-PATH (HP). More precisely, these worst-case instances can be easily reduced from any instance of (HP) showing how they admit a covering route for all the targets if and only if the corresponding instance of HP admits an Hamiltonian path. They are defined as follows:

- all the vertices are targets;
- edge costs are set to 1;
- there is only one signal;
- penetration times are set to $|T| - 1$;
- values are drawn from $(0, 1]$ with uniform probability for all the targets;
- the number of edges is drawn from a normal distribution with mean ϵ, said *edge density* and defined as $\epsilon = |E| / \frac{|T|(|T|-1)}{2}$;
- starting vertex v is drawn among the targets of T with uniform probability.

We explore two parameter dimensions: the number of targets $|T|$ and the value of edge density ϵ. In particular, we use the following values:

$$|T| \in \{6, 8, 10, 12, 14, 16, 20, 25, 30, 35, 40, 45, 50\},$$
$$\epsilon \in \{0.05, 0.10, 0.25, 0.50, 0.75, 1.00\}.$$

For each combination of values of $|T|$ and ϵ, we randomly generate 100 instances with the constraint that, if $\epsilon \frac{|T|^2}{2} < |T|$, we introduce additional edges in order to assure the graph connectivity. The suitability of our worst-case analysis is corroborated by the results obtained with a realistic setting (see Section 5.2), which present hard subproblems characterized by the features listed above.

5.1.2. Exact algorithms scalability

We report in Fig. 5 the compute time (averaged over 100 SRG-v instances) required by our exact dynamic programming algorithm (Algorithm 1), with the annotation of dominated covering sets and the generation of the routes, as $|T|$ and ϵ vary. We report in Appendix C the boxplots showing the statistical significance of the results. It can be observed that the compute times are exponential in $|T|$, the curves being lines in a semilogarithmic plot, and the value of ϵ determines the

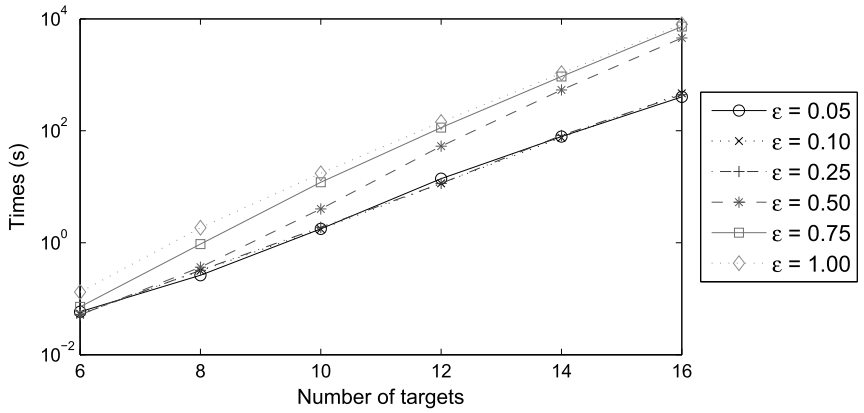


Fig. 5. Compute times in seconds of our exact dynamic programming algorithm (Algorithm 1), with the annotation of dominated covering sets and the generation of the routes, as $|T|$ and ϵ vary.

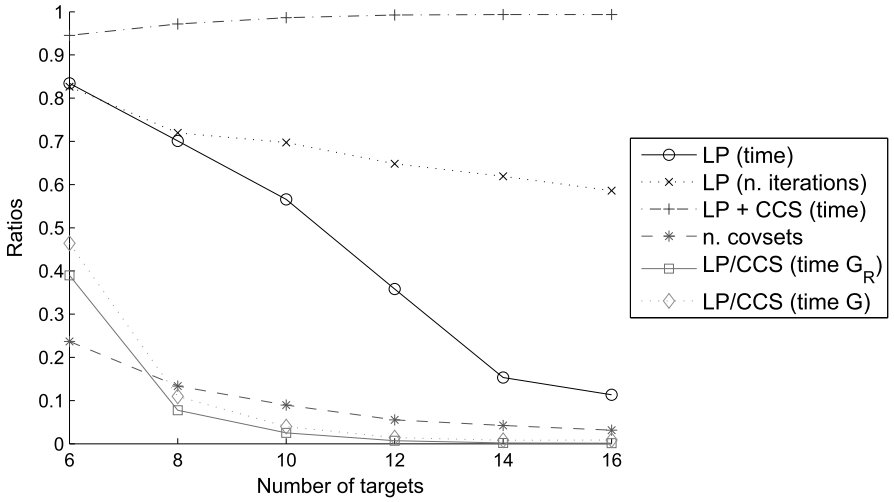


Fig. 6. Ratios evaluating dominances with $\epsilon = 0.25$ as $|T|$ varies.

slope of the line. Notice that with $\epsilon \in \{0.05, 0.10, 0.25\}$ the number of edges is almost the same when $|T| \leq 16$ due to the constraint of connectivity of the graph, leading thus to the same compute times. Beyond 16 targets, the compute times of our exact dynamic programming algorithm are excessively long (with only $\epsilon = 0.25$, the compute time when $|T| = 20$ is lower than 10^4 seconds). Interestingly, the compute time monotonically decreases as ϵ decreases. This is thanks to the fact that the number of covering sets generated by Algorithm 1 dramatically reduces as ϵ reduces.

We do not report any plot of the compute times of our exact branch-and-bound algorithm, since it requires more than 10^4 seconds when $|T| > 8$ even with $\epsilon = 0.25$, resulting thus non-applicable in practice. This is because the branch-and-bound algorithm has a complexity $O(|T|^{|T|})$, while the dynamic programming algorithm has a complexity $O(2^{|T|})$.

Fig. 6 shows the impact of discarding dominated actions from the game when $\epsilon = 0.25$. 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 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 $\mathcal{G}$. As it can be seen, non-dominated actions constitute a small percentage, decreasing with the number of targets. This result indicates that the structure of the problem exhibits a non-negligible degree of redundancy. LP times (iterations) report the ratio between $\mathcal{G}_R$ and $\mathcal{G}$ for the time (iterations) required to solve the maxmin 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, that is LP + 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.

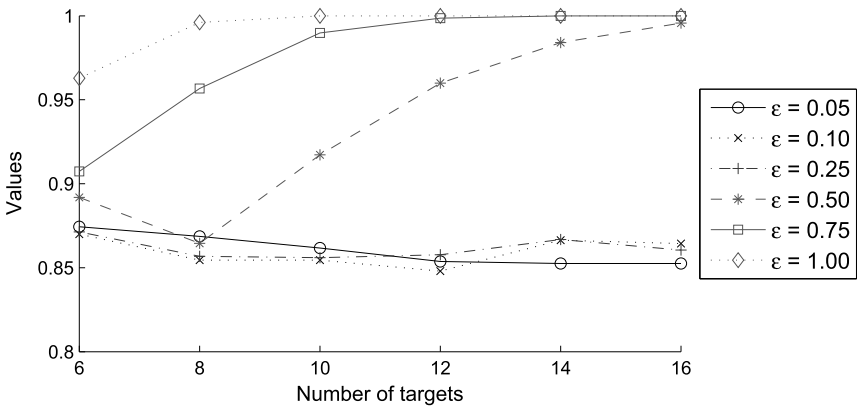


Fig. 7. Optimal game values as $|T|$ and ϵ vary.

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

m	$\|T(s)\|$		
	5	10	15
2	–	17.83	510.61
3	–	33.00	769.30
4	0.55	35.35	1066.76
5	0.72	52.43	1373.32

Fig. 7 shows the game value for $\mathcal{D}$, $1 - g_v$, as $|T|$ and ϵ vary (averaged over 100 instances). It can be observed that the game value is almost constant as $|T|$ varies for $\epsilon \in \{0.05, 0.10, 0.25\}$ and it is about 0.87. This is because all these instances have a similar number of edges, very close to the minimum number necessary for having connected graphs. With a larger number of edges, the game value increases. Interestingly, fixed a value of ϵ, there is a threshold of $|T|$ such that beyond the threshold the game value increases as $|T|$ increases. This suggests that the minimum game value is obtained for connected graphs with the minimum number of edges.

In Table 2, we report compute times 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. 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.

5.1.3. Approximation algorithms

We evaluate the empirical approximation ratios obtained with our approximation algorithms as $(1 - \hat{g}_v)/(1 - g_v)$, 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. We execute our approximation dynamic programming algorithm with a different number, say RandRes, of randomly generated orders from $\{10, 20, 30, 40, 50\}$, in addition to the 3 heuristics discussed in Section 3.2.2. We executed our approximation branch and bound algorithm with constant values of ρ from $\{0.25, 0.50, 0.75, 1.00\}$ (we recall that with $\rho = 1.00$ backtracking is completely disabled).

Fig. 8 and Fig. 9 report the empirical approximation ratios (averaged over 100 instances) obtained with our approximation algorithms for different values of $|T| \in \{6, 8, 10, 12, 14, 16\}$, i.e., the instances for which we know the optimal game value, and $\epsilon \in \{0.05, 0.10, 0.25, 0.50, 0.75, 1.00\}$. We remark that the ratios obtained with the approximation branch-and-bound algorithm for some values of ρ are omitted. This is because the compute time needed by the algorithm is over 10^4 seconds. The algorithm always terminates by the deadline for only $\rho \in \{0.75, 1.00\}$.

We focus on the ratios obtained with the dynamic programming algorithm. Given a value of ϵ, as $|T|$ increases, the ratio decreases up to a given threshold of $|T|$ and then it is a constant. The threshold increases as ϵ decreases, while the constant decreases as ϵ decreases. The value of the constant is high for every ϵ, being larger than 0.8. Although the ratios increase as RandRes increases, it is worth noting that the increase is not very significant, being of the order of 0.05 between 10 RandRes and 50 RandRes. We focus on the ratios obtained with the branch-and-bound algorithm. Given a value of ϵ, as $|T|$ increases, the ratio decreases up to a given threshold of $|T|$ and then it increases approaching a ratio of 1. The threshold increases as ϵ decreases, while the minimum ratio decreases as ϵ decreases. Interestingly, ratios with $\rho = 1.00$ are very close to ratios with $\rho = 0.75$, showing that performing even significant backtracking around the solution found with $\rho = 1.00$ does not lead to a significant improvement of the solution. The solution can be effectively improved only with $\rho = 0.25$, but it is not affordable due to the excessive required compute time. This shows that the heuristic performs very well. Comparing the ratios of the two algorithms, it can be observed that the approximation dynamic programming

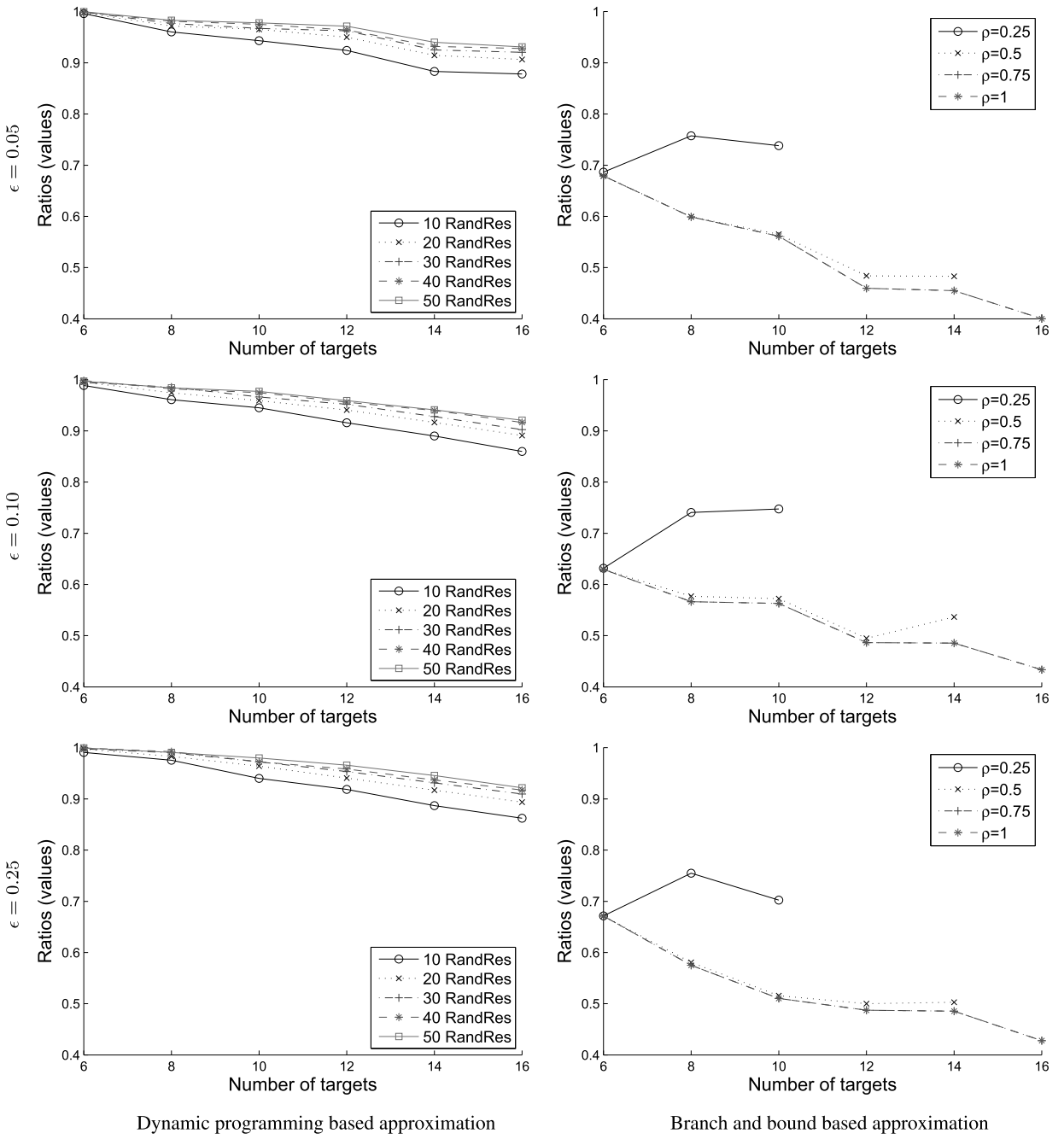


Fig. 8. Approximation ratios as $|T|$ varies.

algorithm performs better than the approximation branch-and-bound algorithm although this last one turned out to be slightly better in a limited number of cases (see Fig. 10). While the dynamic programming one always provides a ratio larger than 0.8, the branch-and-bound one provides for combinations of $|T|$ and ϵ ratios lower than 0.4.

Fig. 10 reports the game values obtained with the approximation dynamic programming algorithm for every value of RandRes and with the approximation branch-and-bound algorithm when $|T| \in \{20, 25, 30, 35, 40, 45, 50\}$ only for $\rho = 1.00$. Indeed, with $\rho = 0.75$ the compute time is excessive and, as shown above, the purely heuristic solution cannot be significantly improved for $\epsilon \geq 0.75$. We report experimental results only for $\epsilon \in \{0.05, 0.25\}$. We notice that for these instances we do not have the optimal game value. However, since the optimal game value cannot be larger than 1 by construction of the instances, the game value obtained with our approximation algorithms represents a lower bound to the actual approximation ratio. It can be observed that, given a value of ϵ, the ratios obtained with the dynamic programming algorithm are

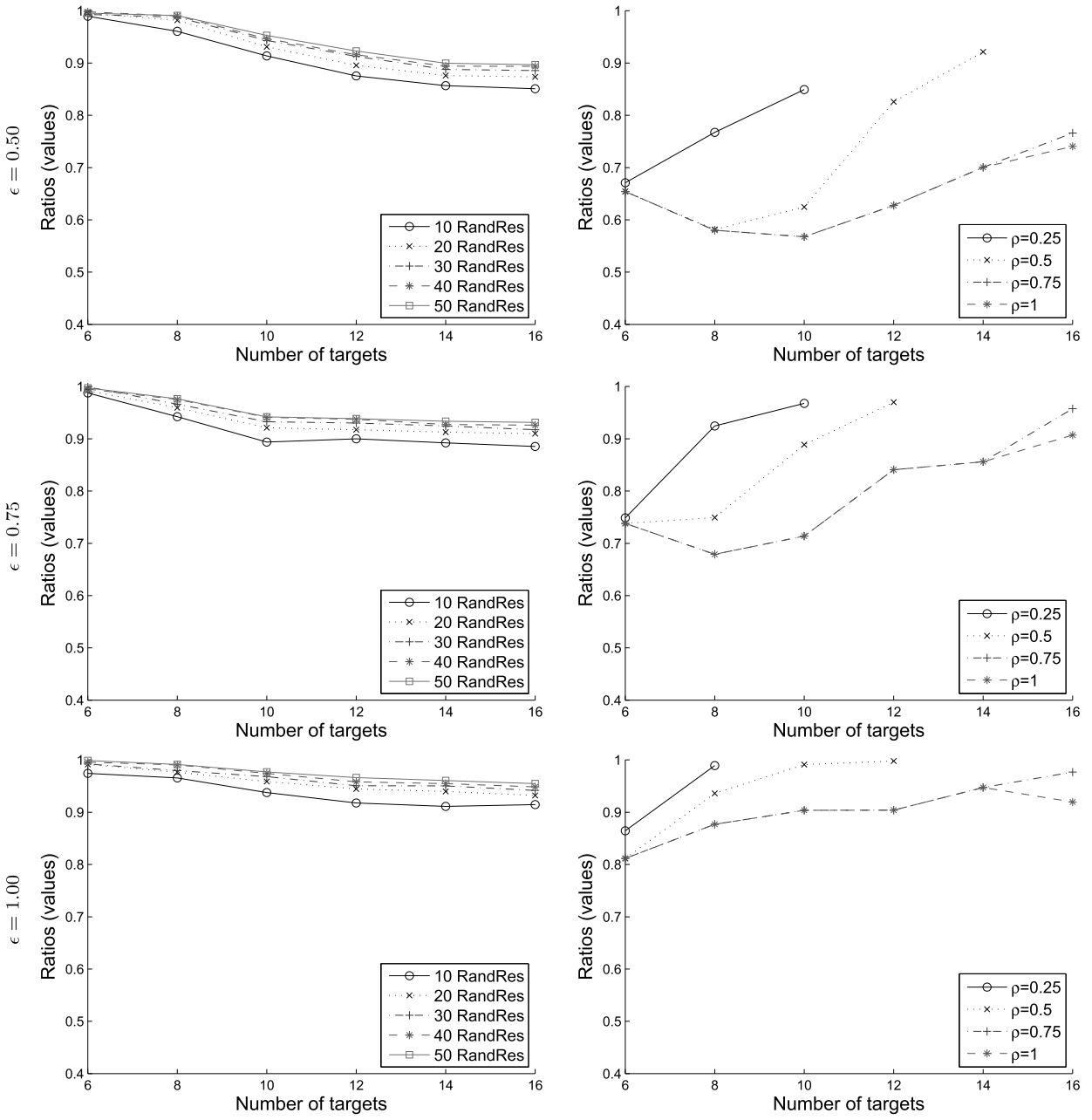


Fig. 9. Approximation ratios as $|T|$ varies.

essentially constant as $|T|$ increases and this constant reduced as ϵ reduces. Surprisingly, after a certain value of $|T|$, the game values obtained with the branch and bound algorithm are higher than those obtained with the dynamic programming algorithm. This is because, fixed a value of ϵ, as $|T|$ increases, the problem becomes easier and the heuristic used by the branch and bound algorithm performs well finding the best covering routes. This shows that there is not an algorithm outperforming the other for every combination of parameters $|T|$ and ϵ. Furthermore, the above result shows that the worst cases for the approximation algorithms are those in which $\epsilon = O(\frac{1}{|T|})$, corresponding to instances in which the number of edges per vertex is a constant in $|T|$. It is not clear from our experimental analysis whether increasing $|T|$ with $\epsilon = \frac{\nu}{|T|}$ for some $\nu > 1$ the game value approaches to 0 or to a strictly positive value. However, our approximation algorithms provide a very good approximation even with a large number of targets and a small value of ϵ.

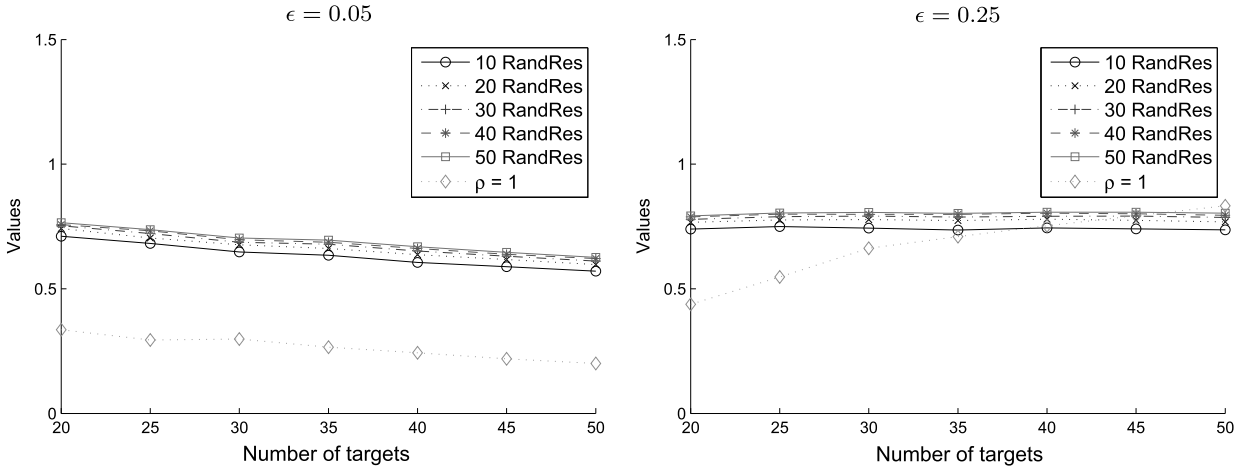


Fig. 10. Game values as $|T|$ varies.

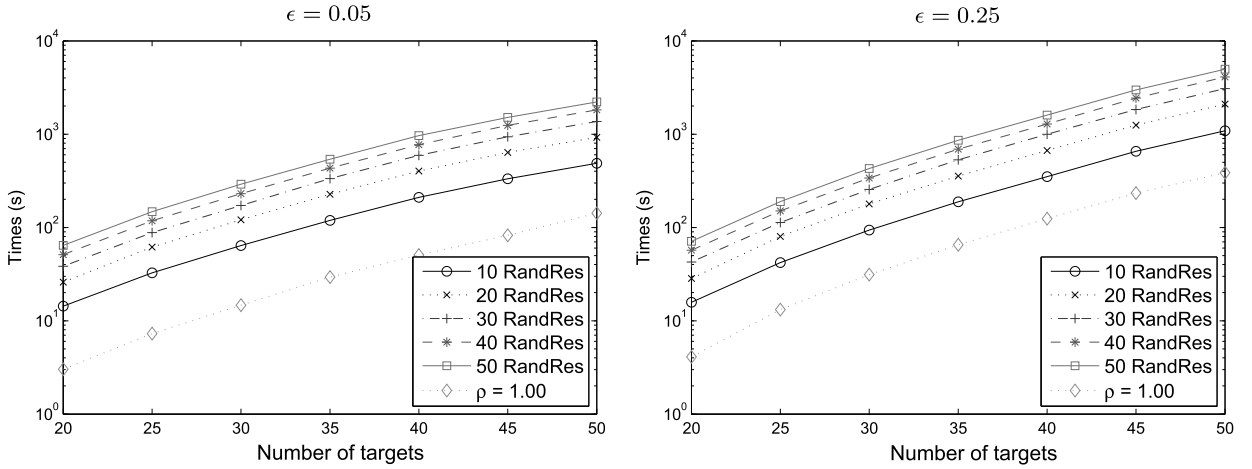


Fig. 11. Compute times as $|T|$ varies.

Fig. 11 reports the compute times required by the approximation dynamic programming algorithms. As it can be seen, the required time slightly increases when adopting a larger number of randomly generated orders with respect to the baseline with $\rho = 1.00$.

5.2. Real case study

In Section 1.1.2 we presented a motivating scenario describing the fair site of Expo 2015. We now formalize an instance for our problem inspired by such scenario to derive important insights on how our techniques operate in real cases. Expo 2015 has been a large setting which, as we anticipated, underwent a remarkable deployment of security resources with about 700 guards operating on the field. This experiment addresses a worst-case scenario where we stress our route-generation algorithms by assuming that each guard has to protect the whole environment. The resolution of this problem is necessary even when admitting multiple defending resources for which no environment partition is pre-assigned (see [27] for a more detailed discussion).

Fig. 12 shows a graph representation of the Expo 2015 site. We manually build a discretized version of the site map by exploiting publicly available knowledge of the event.[9] We identify ≈170 sensible locations which correspond to an equal number of targets in our graph. More specifically, we identify ≈130 targets located at the entrances of each pavilion and in the surroundings of those areas which could be of interest for a high number of visitors. Some targets (≈35) are located over the main roads, being these critical mainly due to their high crowd. Such roads also define our set of edges which resulted in a density of ≈0.02. Fig. 12 reports a graphical representation of chosen deadlines $d(\cdot)$ and values $\pi(\cdot)$, respectively. To

[9] Detailed information can be found at http://www.expo2015.org/ while the map we used for our experiments can be viewed at https://goo.gl/jXPkky.

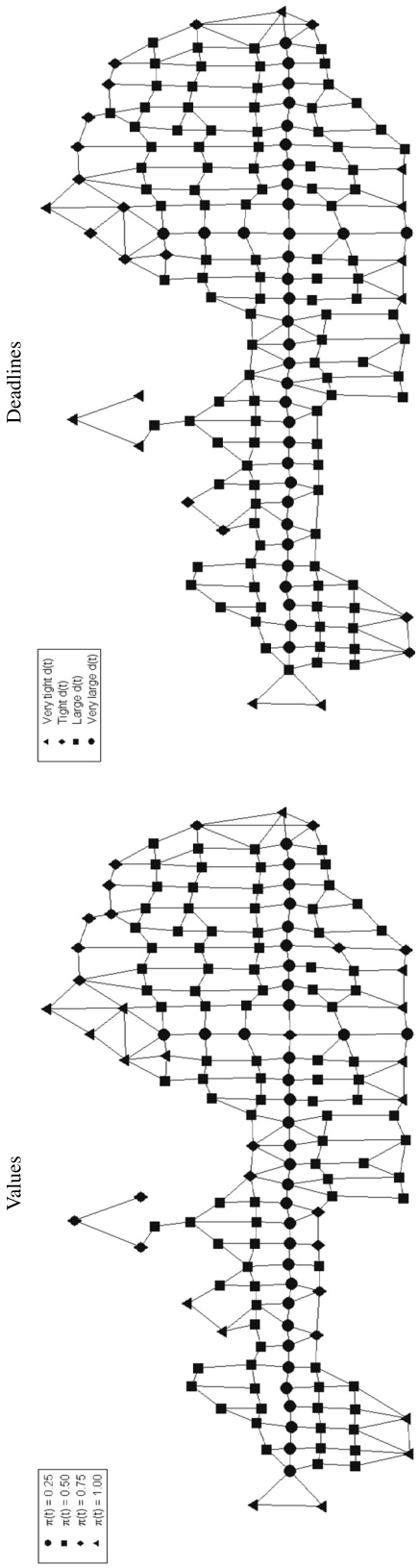


Fig. 12. Expo 2015 instance.

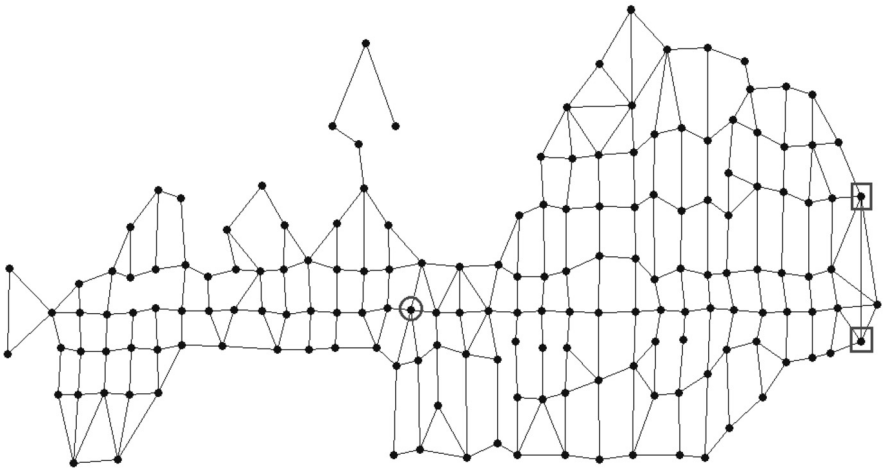
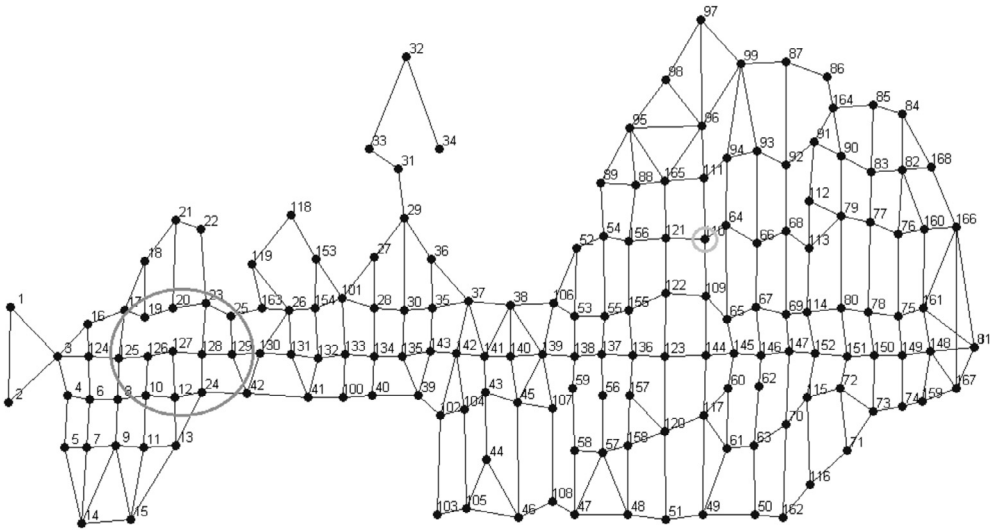


Fig. 13. Best placement and attack locations.

determine such values in a reasonable way we apply a number of simple rules of thumb. First, to ease our task, we discretize the spaces of possible deadlines and values in four different levels. To assign a value to a target, we estimate the interest (in terms of popularity and expected crowd) of the corresponding area in the fair site. The higher the interest, the higher the value (actual values are reported in the figure). To assign deadlines we estimate the time an attacker should spend to escape from the attacked target after some malicious activity is started (for example, blending into the crowd without leaving any trace). In particular, we estimate a smaller escape time for those locations lying near the external border of the fair site while for locations that are more central we estimated a larger time. The smaller the escape time, the tighter the deadline for that target. Actual values are extracted from a normal distribution where $\sigma^2 = 1$ and μ is set according to the chosen level. The maximum distance between any two target locations is about 1.5 km which we assume can be covered in about 7.5 minutes (we imagined a crowded scenario). Given such reference scenario, our means span from 5 minutes (very tight) to 7.5 minutes (very large). To derive our alarm system model we assume to have a number of panoramic cameras deployed in the environment at locations we manually choose in order to cover the whole environment and to guarantee a wide area of view for each camera (i.e., trying to keep, in general, more than one target under each camera's view). To map our set of cameras over the alarm system model, we adopt this convention: each group of cameras sharing an independent partial view of a target t is associated to a signal $s \in S(t)$; if target t is covered by k signals then each signal is generated with probability $1/k$ once t is attacked. Obviously, a deeper knowledge of the security systems deployed on the site can enable specific methods to set the parameters of our model. This is why we encourage involving agencies in charge of security when dealing with such task.

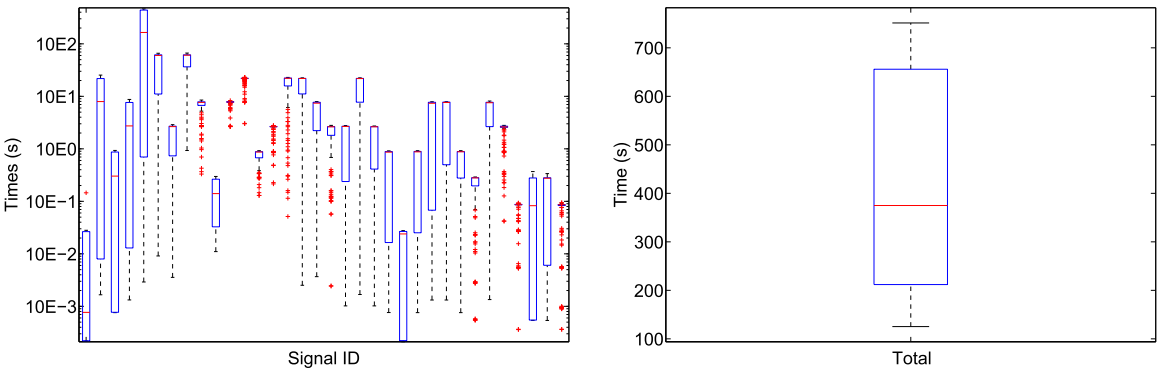
We first show a qualitative evaluation of our method. Fig. 13 depicts the best placement for the Defender (the circle in the figure) and the attacked targets (the squares in the figure, these are the actions played by the Attacker with non-null probability at the equilibrium). As intuition would suggest, the best location from where any signal response should start is a central one w.r.t. the whole fair site. Our simulations show that the optimal patrolling strategy coincides with such fixed placement even under false negatives rates of at least ≈ 0.3. Notice that such false negatives value can be considered unrealistically pessimistic for alarm systems deployed in structured environment like the one we are dealing with. Attacked targets correspond to areas, which exhibit rather high interest and small escape time. Fig. 14 reports an example of signal response strategy for a given starting vertex (the small circle in the figure) and a given signal (whose covered targets are depicted with the large circle in the figure). The table lists the computed covering sets and the probabilities with which the Defender plays the corresponding covering routes.

Boxplots of Fig. 15(a) provide some quantitative insights on the computational effort we measured in solving such realistic instance. Given a signal, we report the statistical distribution of the time required by Algorithm 1 to compute covering routes from each possible start vertex. In general, we observe a high variance in each boxplot. Indeed, once fixed a signal s in our realistic instance, it is easy to identify starting vertices from which computing covering routes is likely to be very easy or, instead, much harder. For the easy case, consider a starting vertex lying very much far away from the group of targets covered by s. In such case, Algorithm 1 will soon stop iterating through covering set cardinalities being not able to further generate feasible sets. Such feature is induced by the large distance of the starting vertex from the targets covered by s together with the low edge density and the spatial locality shared among targets covered by the same signal (these last two are, indeed, features that frequently recur in realistic scenarios). For the harder case, just consider a situation in which the distance of the starting vertex from the targets covered by s is such that a large number of covering routes is available. An example of this kind can be inspected in Fig. 14. Interestingly, a similar high variance trend is not always observed when depicting the statistical distribution of the compute time per starting vertex. The boxplot of Fig. 15(b) shows an example of the distribution of compute time from a given starting vertex (the sample here is the composed by compute



Covering set	Probability
{19, 20, 23, 25, 125, 126, 127, 128}	0.0194
{10, 12, 23, 24, 25, 126, 127, 128}	0.0231
{10, 12, 24, 25, 126, 127, 128, 129}	0.0333
{12, 23, 24, 25, 126, 127, 128, 129}	0.0494
{10, 12, 23, 24, 25, 125, 126, 128}	0.0344
{10, 12, 24, 25, 125, 126, 128, 129}	0.0488
{10, 12, 25, 125, 126, 127, 128, 129}	0.0493
{12, 23, 24, 25, 125, 126, 127, 128}	0.0502
{12, 24, 25, 125, 126, 127, 128, 129}	0.0692
{19, 20, 23, 25, 125, 126, 129}	0.0492
{19, 20, 23, 125, 126, 128, 129}	0.0492
{20, 23, 25, 126, 127, 128, 129}	0.0657
{10, 23, 25, 125, 126, 127, 128}	0.0662
{19, 20, 23, 25, 125, 128, 129}	0.0412
{23, 25, 125, 126, 127, 128, 129}	0.1146
{20, 23, 24, 25, 127, 128}	0.0645
{10, 12, 24, 125, 126, 127}	0.0877
{20, 23, 24, 25, 128, 129}	0.0846

Fig. 14. Example of response strategies to signal.



(a) Time boxplots by signal (b) Time boxplots by node

Fig. 15. Time boxplots for our real case study.

times associated to the resolution from that starting vertex with respect to each signal). Such distribution (characterized by a lower variance with respect to what can be observed in Fig. 14) suggests that there can be starting vertices posing easy resolution and, similarly, others posing only hard ones.

6. Related works

In the last few years, Security Games received an increasing interest from the Artificial Intelligence scientific community, leading to the exploration of a large number of research directions around this topic. In this section, we briefly discuss what we deem to be the most significant ones, starting from the game theoretical foundations on which these models are built.

Computing *solution concepts* is the central problem upon which the real applicability of these game theoretical models is based. A lot of works concentrated on algorithmic studies of this topic, analysing the relationships holding among different kinds of solution concepts and their computational complexity. In [39] the relationship between Stackelberg, Nash and min–max equilibria is studied, while in [40] some refinements of the Stackelberg equilibrium are proposed. Many efforts have been made to develop tractable algorithms for finding Stackelberg equilibria in Bayesian games [41]. Furthermore, in [42], the authors analysed scenarios in which the Defender has multiple objectives, searching for the Pareto curve of the Stackelberg equilibria.

Besides fundamental works like the ones cited above, a more recent research line devoted efforts towards the definition of game model refinements in the attempt to overcome some of their ideal assumptions. One remarkable issue belonging to this scope is how to model the behavior of the Attacker. In the attempt to have a more realistic behavior, some works considered *bounded rationality* and defined algorithms to deal with it. In [43] different models of the Attacker are analysed while in [44,45] the Attacker is allowed to have different observation and planning capabilities. Moreover, in [46] Quantal–Best Response is used to model the behavior of the Attacker and in [47] algorithms that scale up with bounded rational adversaries are proposed. In our paper, we assume that the attacker is rational.

Other model refinements focused on those cases in which games exhibit *specific structures* that can be leveraged in the design of algorithms to compute the Stackelberg equilibrium. For instance, the study of the spread of contagion over a network is investigated in [48]. When no scheduling constraints are present and payoffs exhibit a special form, the computation of a Stackelberg equilibrium can be done very efficiently enabling the resolution of remarkably big scenarios [49]. In [50], realistic aspects of infrastructures to be protected are taken into account.

7. Conclusions and future research

In this paper we provide the first Security Game for large environments surveillance, e.g., for fair sites protection, that can exploit an alarm system with spatially uncertain signals. To monitor and protect large infrastructure such as stations, airports, and cities, a two-level paradigm is commonly adopted: a broad area surveillance phase, where an attack is detected but only approximately localized due to the spatially uncertainty of the alarm system, triggers a local investigation phase, where guards have to find and clear the attack. Abstracting away from technological details, 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 FNP-hard on trees and APX-hard with arbitrary graphs. Then, we provide two exponential-time exact algorithms to find the best patrolling strategy to respond to a given alarm signal. The first algorithm performs a breath-first search by exploiting a dynamic programming approach, while the second algorithm performs a depth-first approach by exploiting a branch-and-bound approach. We provide also a variation of these two algorithms to find an approximate solution. We experimentally evaluate our exact and approximation algorithms both in worst-case instances, to evaluate empirically the gap between our hardness results and the theoretical guarantees of our approximation algorithms, and in one realistic instance. The limit of our exact algorithms is about 16 targets with worst-case instances while we were able to compute an optimal solution for a realistic instance with ≈ 170 targets. On the other side, our approximation algorithms provide a very effective approximation even with worst-case instances. Finally, we focus on the problem of patrolling the environment, showing that if every target is alarmed and no false positives nor missed detections are present, then the best patrolling strategy prescribes that the patroller stays in a given place waiting for an alarm signal. Furthermore, we show that such a strategy may be optimal even for missed detection rates up to 50%.

Besides the solutions we studied, a number of open problems have been posed for future research. The main theoretical issue is the closure of the approximation gap of SRG-v. We believe that investigating the relationship between our model and the DEADLINE–TSP could help in closing the gap. Another interesting problem is the study of approximation algorithms for tree graphs. Our NP-hardness result does not exclude the existence of a PTAS (i.e., polynomial time approximation scheme), even if we conjecture that the existence is unlikely. A number of extensions of our model are worth being explored, we identify two prominent ones. The first is to enrich our model with false positives and missed detections, requiring patrolling even in the absence of alarm signals. A second extension is to consider settings with multiple patrollers. We deem that the techniques presented in this work can play a fundamental role in scaling to multi-patroller settings. In [27] we give some preliminary contributions along this direction, discussing different arising subproblems, showing how some properties like the one given in Theorem 10 can be generalized, and leveraging the algorithms presented here for the resolution of multi-patroller games under different coordination assumptions.

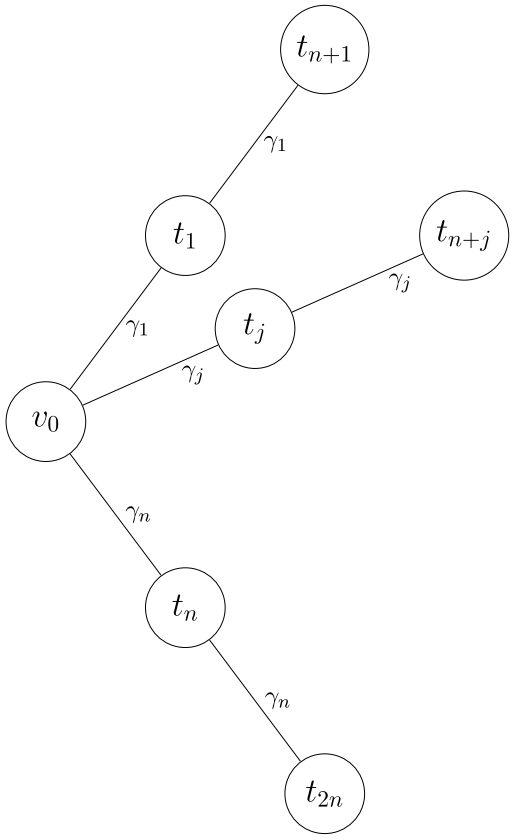


Fig. A.16. Special 2-level star graph.

Appendix A. Proofs

A.1. Proof of Theorem 1

We first define a special class of trees we call *special 2-level stars* (S2L–STAR). See Fig. A.16 for an example.

Definition 11 *(S2L–STAR).* Special 2-level star graph instances (S2L–STAR) are:

- $V = \{v_0, v_1, v_2, \ldots, v_{2n}\}$, where v_0 is the starting position;
- $T = V \setminus \{v_0\}$, where vertices v_i with $i \in \{1, \ldots, n\}$ are called *inner targets*, while vertices v_i with $i \in \{n+1, \ldots, 2n\}$ are called *outer targets*;
- $E = \{(v_0, v_i), (v_i, v_{n+i}) : \forall i \in \{1, \ldots, n\}\}$ and we refer to the pair of edges $((v_0, v_i), (v_i, v_{n+i}))$ as the i-th branch;
- travel costs are $c(v_0, v_i) = c(v_i, v_{n+i}) = \gamma_i$ for every $i \in \{1, \ldots, n\}$, where $\gamma_i \in \mathbb{N}^+$;
- penetration times are, for $i \in \{1, \ldots, n\}$, $d(t) = \begin{cases} 6H - 3\gamma_i & t = v_i \\ 10H - 2\gamma_i, & t = v_{n+i} \end{cases}$, where $H = \frac{\sum_{i=1}^{n} \gamma_i}{2}$;
- $\pi(t) = 1$ for every $t \in T$.

Initially, we show a property of S2L–STAR instances that we shall use below.

Lemma 12. *If an instance of S2L–STAR admits a maximal covering route r that covers all the targets, then the branches can be partitioned in two sets C_1 and C_2 such that:*

- *all the branches in C_1 are visited only once while all the branches in C_2 are visited twice;*
- $\sum_{i \in C_1} \gamma_i = \sum_{i \in C_2} \gamma_i = H$.

Proof. Initially, we observe that, in a feasible solution, the visit of a branch can be of two forms. If branch i is visited once, then $\mathcal{D}$ will visit the inner target before time $6H - 3\gamma_i$ and immediately after the outer target. C_1 denotes the set of all the

branches visited according to this form. If branch i is visited twice, then $\mathcal{D}$ will visit at first the inner target before time $6H - 3\gamma_i$, coming back to v_0 immediately after, and subsequently in some time after $6H - 3\gamma_i$, but before $10H - 2\gamma_i$, $\mathcal{D}$ will visit again the inner target and immediately after the outer target. C_2 denotes the set of all the branches that are visited according to this form. All the other forms of visits (e.g., three or more visits, different order visits, and visits at different times) are useless and any route in which some branch is not neither in C_1 nor in C_2 can be modified such that all the branches are either in C_1 or in C_2 strictly decreasing the cost of the solution as follows:

- if branch i is visited only once and the visit of the inner target is after time $6H - 3\gamma_i$, then the solution is not feasible;
- if branch i is visited twice and the first visit of the inner target is after time $6H - 3\gamma_i$, then the solution is not feasible;
- if branch i is visited twice and the second visit of the inner target is before time $6H - 3\gamma_i$, then the first visit of the branch can be omitted saving $2\gamma_i$;
- if branch i is visited twice and the outer target is visited during the first visit, then the second visit of the branch can be omitted saving $\geq 2\gamma_i$;
- if branch i is visited three or more times, all the visits except the first one in which the inner target is visited and the first one in which the outer target is visited can be omitted saving $\geq 2\gamma_i$.

We assume that, if there is a maximal covering route r that covers all the targets, then the visits are such that $C_1 \cup C_2 = \{1, \ldots, n\}$ and therefore that each branch is visited either once or twice as discussed above. We show below that in S2L–STAR instances such an assumption is always true. Since r covers all the targets, we have that the following conditions are satisfied:

$$2\sum_{i \in C_2} \gamma_i + 4\sum_{i \in C_1} \gamma_i \leq 6H \tag{A.1}$$

$$6\sum_{i \in C_2} \gamma_i + 4\sum_{i \in C_1} \gamma_i \leq 10H \tag{A.2}$$

Constraint (A.1) requires that the cost of visiting entirely all the branches in C_1 and partially (only the inner target) all the branches in C_2 is not larger than the penetration times of the inner targets. Notice that this holds only when the last inner target is first-visited on a branch in C_1. We show below that such assumption is always verified. Constraint (A.2) requires that the cost of visiting entirely all the branches in C_1 and at first partially and subsequently entirely all the branches in C_2 is not larger than the penetration times of the outer targets. We can simplify the above pair of constraints as follows:

$$\underbrace{2\sum_{i \in C_2} \gamma_i + 2\sum_{i \in C_1} \gamma_i}_{4H} + 2\sum_{i \in C_1} \gamma_i \leq 6H$$

$$2\sum_{i \in C_2} \gamma_i + \underbrace{4\sum_{i \in C_2} \gamma_i + 4\sum_{i \in C_1} \gamma_i}_{8H} \leq 10H$$

obtaining:

$$\sum_{i \in C_1} \gamma_i \leq H$$

$$\sum_{i \in C_2} \gamma_i \leq H$$

since, by definition, $\sum_{i \in C_1} \gamma_i + \sum_{i \in C_2} \gamma_i = 2H$, it follows that:

$$\sum_{i \in C_1} \gamma_i = \sum_{i \in C_2} \gamma_i = H.$$

Therefore, if r covers all the targets and it is such that all the branches belong either to C_1 or to C_2, we have that r visits the last outer target exactly at its penetration time. This is because Constraints (A.1) and (A.2) hold as equalities. Thus, as shown above, in any route in which a branch is neither in C_1 nor in C_2, we can change the visits such that all the branches are in either C_1 or C_2, strictly reducing the total cost. It follows that no route with at least one branch that is not neither in C_1 nor in C_2 can have a total cost equal to or smaller than the penetration time of the outer targets. Similarly, from the above equality it follows that any solution where the last inner target is first-visited on a C_2 branch can be strictly improved

by moving such branch to C_1 and therefore no route in which the last inner target is first-visited on a C_2 branch can have a total cost equal to or smaller than the penetration time of the outer targets. □

We can now prove Theorem 1.

Proof. We provide a reduction from PARTITION that is known to be weakly NP-hard.

Definition 12 *(PARTITION).* The decision problem PARTITION is defined as:
INSTANCE: A finite set $I = \{1, 2, \dots, l\}$, a size $a_i \in \mathbb{N}^+$ for each $i \in I$, and a bound $B \in \mathbb{N}^+$ such that $\sum_{i \in I} a_i = 2B$.
QUESTION: Is there any subset $I' \subseteq I$ such that $\sum_{i \in I'} a_i = \sum_{i \in I \setminus I'} a_i = B$?

For the sake of clarity, we divide the proof in steps.
Reduction. We map an instance of PARTITION to an instance of k–SRG-v on S2L–STAR graphs as follows

- $S = \{s\}$,
- $n = l$ (i.e., the number of branches in S2L–STAR equals the number of elements in PARTITION);
- $\gamma_i = a_i$ for every $i \in I$;
- $H = B$,
- $k = 0$.

The rationale is that there is a feasible solution for PARTITION if and only if there is the maximal covering route that covers all the targets in a k–SRG-v on a S2L–STAR graph.

If. From Lemma 12 we know that, if there is the maximal covering route that covers all the targets in a k–SRG-v on a S2L–STAR graph, then the branches can be partitioned in two sets C_1, C_2 such that $\sum_{i \in C_1} \gamma_i = \sum_{i \in C_2} \gamma_i = H$. By construction $\gamma_i = a_i$ and $H = B$. So, if there is the maximal covering route that covers all the targets in a k–SRG-v on a S2L–STAR graph, then there is partition of set I in two subsets $I' = C_1$ and $I'' = C_2$ such that $\sum_{i \in C_1} \gamma_i = \sum_{i \in I'} a_i = H = B = \sum_{i \in C_2} \gamma_i = \sum_{i \in I''} a_i$.

Only if. If PARTITION admits a feasible solution, then, once assigned $I' = C_1$ and $I'' = C_2$, it is straightforward to see that the route visits all the targets by their penetration times and therefore that the route is a maximal covering route. □

As a final remark, let us notice that this result does not exclude the existence of an FPTAS, i.e., Fully Polynomial Time Approximation Scheme (in fact, PARTITION admits an FPTAS).

A.2. Proof of Theorem 2

The theorem immediately follows from the same proof given for Theorem 1 and reported in the previous section.

A.3. Proof of Theorem 3

Proof. Assume that, for simplicity, $S = \{s_1\}$ and that $T(s_1) = T$. Initially, we observe that MAX–COV-SET is in co-NP. Indeed, any covering route r such that $T(r) \supset T'$ is a NO certificate for MAX–COV-SET, placing it in co-NP. (Notice that, trivially, any covering route has length bounded by $O(|T|^2)$; also, 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 $\mathsf{P} = \mathsf{NP}$.)

Let us suppose we have a polynomial-time algorithm for MAX–COV-SET, called A. Then (since $\mathsf{P} \subseteq \mathsf{NP} \cap \text{co-NP}$) 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 $t_1, t_2, \dots, t_{|T|}$;
2. for every $t \in T$, verify if $\{t\}$ is a covering set in $O(|T|)$ 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 $\mathsf{P} \neq \mathsf{NP}$) would contradict Theorem 2. □

A.4. Proof of Theorem 4

Proof. We produce an approximation-preserving reduction from TSP(1, 2) that is known to be APX-hard [51]. For the sake of clarity, we divide the proof in steps.

TSP(1, 2) instance. An instance of undirected TSP(1, 2) is defined as follows:

- a set of vertices V_{TSP};
- a set of edges composed of an edge per pair of vertices;
- a symmetric matrix C_{TSP} of weights, whose values can be 1 or 2, each associated with an edge and representing the cost of the shortest path between the corresponding pair of vertices.

The goal is to find the minimum cost tour. Let us denote by $OPTSOL_{TSP}$ and OPT_{TSP} the optimal solution of TSP(1, 2) and its cost, respectively. Furthermore, let us denote by $APXSOL_{TSP}$ and APX_{TSP} an approximate solution of TSP(1, 2) and its cost, respectively. It is known that there is no polynomial-time approximation algorithm with $APX_{TSP}/OPT_{APX} < \alpha$ for some $\alpha > 1$, unless $\mathsf{P} = \mathsf{NP}$ [51].

Reduction. We map an instance of TSP(1, 2) to a specific instance of SRG-v as follows:

- there is only one signal s;
- $T(s) = V_{TSP}$;
- $w^*_{t,t'} = C_{TSP}(t, t')$, for every $t, t' \in T(s)$;
- $\pi(t) = 1$, for every $t \in T(s)$;
- $w^*_{v,t} = 1$, for every $t \in T(s)$;
- $d(t) = \begin{cases} OPT_{TSP} & \text{if } OPT_{TSP} = |V_{TSP}| \\ OPT_{TSP} - 1 & \text{if } OPT_{TSP} > |V_{TSP}| \end{cases}$, for every $t \in T(s)$.

In this reduction, we use the value of OPT_{TSP} even if there is no polynomial-time algorithm solving exactly TSP(1, 2), unless $\mathsf{P} = \mathsf{NP}$. We show below that with an additional polynomial-time effort we can deal with the lack of knowledge of OPT_{TSP}.

OPT–SRG-v optimal solution. By construction of the SRG-v instance, there is a covering route starting from v and visiting all the targets $t \in T(s)$, each within its penetration time. This route has a cost of exactly $d(t)$ and it is $\langle v, t_1, \ldots, t_{|T(s)|} \rangle$, where $\langle t_1, \ldots, t_{|T(s)|}, t_1 \rangle$ corresponds to $OPTSOL_{TSP}$ with the constraint that $w^*_{t_{|T(s)|}, t_1} = 2$ if $OPT_{TSP} > |V_{TSP}|$ (essentially, we transform the tour in a path by discarding one of the edges with the largest cost). Therefore, the optimal solution of SRG-v, say $OPTSOL_{SRG}$, prescribes to play the maximal route with probability one and the optimal value, say OPT_{SRG}, is 1.

OPT–SRG-v approximation. Let us denote by $APXSOL_{SRG}$ and APX_{SRG} an approximate solution of OPT–SRG-v and its value, respectively. We assume there is a polynomial-time approximation algorithm with $APX_{SRG}/OPT_{SRG} \geq \beta$ where $\beta \in (0, 1)$. Let us notice that $APXSOL_{SRG}$ prescribes to play a polynomially upper bounded number of covering routes with strictly positive probability. We introduce a lemma that characterizes such covering routes.

Lemma 13. *The longest covering route played with strictly positive probability in $APXSOL_{SRG}$ visits at least $\beta|T(s)|$ targets.*

Proof. Assume by contradiction that the longest route visits $\beta|T(s)| - 1$ targets. The best case in terms of maximization of the value of OPT–SRG-v is, due to reasons of symmetry (all the targets have the same value), when there is a set of $|T(s)|$ covering routes of length $\beta|T(s)| - 1$ such that each target is visited exactly by $\beta|T(s)| - 1$ routes. When these routes are available, the best strategy is to randomize uniformly over the routes. The probability that a target is covered is $\beta - \frac{1}{|T(s)|}$ and therefore the value of APX_{SRG} is $\beta - \frac{1}{|T(s)|}$. This leads to a contradiction, since the algorithm would provide an approximation strictly smaller than β. □

TSP(1, 2) approximation from OPT–SRG-v approximation. We use the above lemma to show that we can build a $(3 - 2\beta)$-approximation for TSP(1, 2) from a β-approximation of OPT–SRG-v. Given an $APXSOL_{SRG}$, we extract the longest covering route played with strictly positive probability, say $\langle v, t_1, \ldots, t_{\beta|T(s)|} \rangle$. The route has a cost of at most $d(t)$, it would not cover $\beta|T(s)|$ targets otherwise. Any tour $\langle t_1, \ldots, t_{\beta|T(s)|}, t_{\beta|T(s)|+1}, \ldots, t_{|T(s)|}, t_1 \rangle$ has a cost not larger than $d(t) - 1 + 2(1 - \beta)|T(s)| = OPT_{TSP} - 1 + 2(1-\beta)|V_{TSP}|$ (under the worst case in which all the edges in $\langle t_{\beta|T(s)|}, t_{\beta|T(s)|+1}, \ldots, t_{|T(s)|}, t_1 \rangle$ have a cost of 2). Given that $OPT_{TSP} \geq |V_{TSP}|$, we have that such a tour has a cost not larger than $OPT_{TSP} - 1 + 2(1-\beta)|V_{TSP}| \leq OPT_{TSP}(3 - 2\beta)$. Therefore, the tour is a $(3 - 2\beta)$-approximation for TSP(1, 2). Since TSP(1, 2) is not approximable in poly-

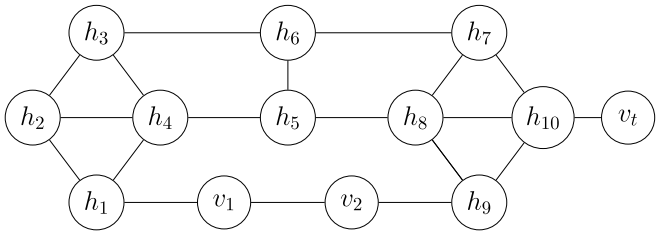


Fig. A.17. Example of construction used in the proof of Theorem 11: the Hamiltonian path $\langle h_1, h_2, h_3, h_4, h_5, h_6, h_7, h_8, h_9, h_{10}\rangle$ on G_H is given, as well as covering route r_1 with $r_1(0) = v_1$ and $T(r_1) = T$. It can be observed that there is another Hamiltonian path for G_H, i.e., $\langle h_9, h_8, h_5, h_4, h_1, h_2, h_3, h_6, h_7, h_{10}\rangle$, allowing covering route r_2 with $r_2(0) = v_2$ and $T(r_2) = T$. Notice that, if we remove the edge (h_5, h_8), then covering route r_2 such that $r_2(0) = v_2$ and $T(r_2) = T$ does not exist.

nomial time for any approximation ratio smaller than α, we have the constraint that $3-2\beta \geq \alpha$, and therefore that $\beta \leq \frac{3-\alpha}{2}$. Since $\alpha > 1$, we have that $\frac{3-\alpha}{2} < 1$ and therefore that there is no polynomial-time approximation algorithm for OPT–SRG-v when $\beta \in (\frac{3-\alpha}{2}, 1)$, unless $\mathsf{P} = \mathsf{NP}$.

OPT_{TSP} oracle. In order to deal with the fact that we do not know OPT_{TSP}, we can execute the approximation algorithm for OPT–SRG-v using a guess over OPT_{TSP}. More precisely, we execute the approximation algorithm for every value in $\{|V_{TSP}|, \ldots, 2|V_{TSP}|\}$ and we return the best approximation found for TSP(1, 2). Given that $OPT_{TSP} \in \{|V_{TSP}|, \ldots, 2|V_{TSP}|\}$, there is an execution of the approximation algorithm that uses the correct guess. □

A.5. Proof of Theorem 11

Proof. We divide the proof in two steps, membership and hardness.

Membership. Given a YES certificate constitutes by a route, the verification is easy, requiring one to apply the route and check whether each target is visited by its deadline. It requires linear time in the number of targets.

Hardness. Let us consider the Restricted Hamiltonian Circuit problem (RHC) which is known to be NP-complete. RHC is defined as follows: given a graph $G_H = (V_H, E_H)$ and an Hamiltonian path $P = \langle h_1, \ldots, h_n\rangle$ for G_H such that $h_i \in V_H$ and $(h_1, h_n) \notin E_H$, find an Hamiltonian cycle for G_H. From such instance of RHC, following the approach of [38], we build the following instance for LM–COV–ROUTE:

- $V = V_H \cup \{v_1, v_2, v_t\}$;
- $T = V_H \cup \{v_t\}$;
- $E = E_H \cup \{(h_n, v_t), (h_i, v_s) : i \in \{1, \ldots, n\}\}$;
- $d(v_t) = n + 1$ and $d(t) = n$ for any $t \in T$ with $t \neq v_t$;
- $w_{v,v'} = \begin{cases} 1 & \text{if } v = h_n, v' = v_t \\ 1 & \text{if } v = h_i, v' = h_j, \forall i, j \in \{1, \ldots n\} \\ 1 & \text{if } v = v_1, v' = h_1 \\ 2 & \text{if } v = v_1, v' = h_{n-1} \\ \geq 2 & \text{if } v = v_1, v' = h_i, \forall i \in \{1, \ldots n-2, n\} \\ \geq 2 & \text{if } v = v_1, v' = v_t \\ 2 & \text{if } v = v_2, v' = h_1 \\ 1 & \text{if } v = v_2, v' = h_{n-1} \\ \geq 2 & \text{if } v = v_2, v' = h_i, \forall i \in \{1, \ldots n-2, n\} \\ \geq 2 & \text{if } v = v_2, v' = v_t \end{cases}$;
- $r_1 = \langle v_1, h_1, \cdots, h_n, v_t\rangle$.

Basically, given G_H, we introduce three vertices v_1, v_2, v_t, where v_1, v_2 are adjacent starting vertices and v_t is a target. We know the covering routes from v_1, and we aim at finding the covering routes from v_2. The new starting vertex (v_2) is closer to h_{n-1} than the previous one (v_1) by 1 and farther from h_1 than previous one (v_1) by 1. There is no constraint over the distances between the starting vertices and the other targets except that they are larger than or equal to 2. We report in Fig. A.17 an example of the above construction. Notice that, by construction, if the maximal covering route r_2 with $r_2(0) = v_2$ and $T(r_2) = T$ exists, then v_t must be the last visited target. Route r_1 is covering since $\langle h_1, \ldots, h_n\rangle$ is a Hamiltonian path for G_H. We need to show that route r_2 with $r_2(0) = v_2$ and $T(r_2) = T$ exists if and only if G_H admits a Hamiltonian cycle. It can observed that, if r_2 exists, then it must be such that $r_2 = \langle v_2, h_{n-1}, \ldots, h_n, v_t\rangle$ and therefore $\langle h_{n-1}, \ldots, h_n\rangle$ must be

a Hamiltonian path for G_H. Since we know, by r_1, that $(h_{n-1}, h_n) \in E_H$, it follows that $\langle h_{n-1}, \ldots, h_n, h_{n-1} \rangle$ is a Hamiltonian cycle. □

Appendix B. Additional results for special topologies

In this section we show that for some special classes of graphs we can give a polynomial-time algorithm for answering Question 2.

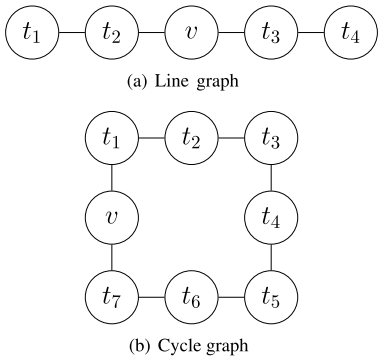
B.1. Line and cycle graphs

Let us first concentrate on line graphs for which an example is depicted in Fig. B.18(a). Consider a modified version of Algorithm 1 where, when deciding on admissible expansions, we add to Conditions 1–3 the following one: the cost of the newly formed covering set must be smaller than or equal to the penetration time of any target not included in such set. Given $Q^k_{v,t}$, the expansion $Q^{k+1}_{v,w}$ is admissible only if for any $t \in T(s) \setminus Q^k_{v,w}$ the inequality $d(t) \geq c(Q^{k+1}_{v,w})$ holds. Such modified algorithm runs in polynomial time on line graph instances. To see this, notice that any covering set enumerated has only two possible expansions and can be compactly represented by two vertices: a left one and a right one. All the targets between the two extremes are covered by the covering set. This makes the number of expansions performed by the algorithm $O(|T(s)|^2)$.

The soundness of the algorithm can simply be determined by noticing that the new condition we imposed on admissible expansions prevents Algorithm 1 from generating only those that would never led to the covering set including all the targets. This implies that we can answer Question 2 in polynomial time for this class of instances. Notice that this results can be easily extended to cycle graphs, see Fig. B.18(b) and to tree graphs where the number of leaves is fixed.

Assessing the computational complexity of finding the maxmin equilibrium in line graphs is a problem left open in this paper. While when searching for a route covering all targets we can safely consider only covering sets defined by two extremes, this is not longer true when searching for an equilibrium. The main reason is that we cannot safely discard routes traversing targets whose penetration time is expired and this does not allow one to provide a polynomially bounded representation of the covering sets. We conjecture that the problem is FNP-hard. To support our conjecture, we provide a class of instances whose non-dominated covering sets rises exponentially in $|T|$ for the values of $|T|$ such that the problem can be solved in reasonable time. More precisely, we empirically observed that the number of non-dominated covering sets of these instances is $\Omega(2^{\frac{|T|-1}{2}})$ for $|T| \leq 23$. With $|T| \geq 25$, Algorithm 1 requires too long time to solve the problem and therefore we could not evaluate the number of non-dominated covering sets (a formal proof that the number of covering sets is exponential does not seem straightforward). While this is not a hardness proof, we believe that it could be an evidence and we report such instances because they could be useful for any researcher interested in the study of line graphs. The instances are defined as follow: given the starting vertex v, there are $(|T|-1)/2$ targets at the left of v and $(|T|-1)/2$ targets at the right of v. The cost of each edge is 1. The value of $d(t)$ is the square of the cost of the shortest path between t and v. In these instances, there are non-dominated routes continuously oscillating from the left to the right of the starting vertex v and *vice versa*.

Let us now consider a *star graph*, as shown in Fig. B.19, defined as follows.



(a) Line graph

(b) Cycle graph

Fig. B.18. Special graphs: line and cycle.

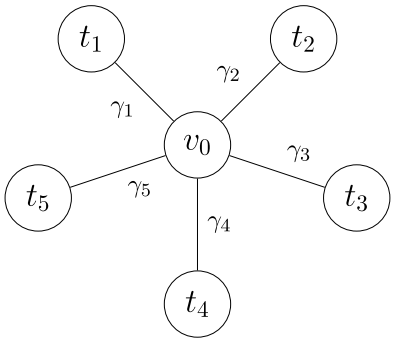


Fig. B.19. Star graph.

Definition 13 *(SIMPLE–STAR).* Simple star graph instances (SIMPLE–STAR) are:

- $V = \{v_0, v_1, v_2, \ldots, v_n\}$, where v_0 is the starting vertex of $\mathcal{D}$;
- $T = V \setminus \{v_0\}$;
- $E = \{(v_0, v_i), \forall i \in \{1, \ldots, n\}\}$;
- travel costs are $c(v_0, v_i) = \gamma_i$, where $\gamma_i \in \mathbb{N}^+$;
- penetration times $d(t)$ and values $\pi(t)$ can be any.

We can state the following theorem, whose proof is not direct as in the case of line and cycle graphs and thus more details are necessary.

Theorem 14. *If the maximal covering route r covering all the targets exists, the Earliest Due Date algorithm returns r in polynomial time once applied to SIMPLE–STAR graph instances.*

Proof. The Earliest Due Date [52] (EDD) algorithm is an optimal algorithm for synchronous (i.e., without release times) aperiodic scheduling with deadlines. It executes (without preemption) the tasks in ascending order according to the deadlines, thus requiring polynomial complexity in the number of tasks. Any SIMPLE–STAR graph instance can be easily mapped to a synchronous aperiodic scheduling problem: each target t_i is an aperiodic task J_i, the computation time of J_i is equal to $2\gamma_i$, the deadline of task J_i is $d(t_i) + \gamma_i$. It is straightforward to see that, if EDD returns a feasible schedule, then there is the maximal covering route, and, if EDD returns a non-feasible schedule, then there is not any maximal covering route. □

The above result shows that Question 2 can be answered in polynomial time.

We focus on Question 3, showing that also this question can be answered in polynomial time for the above special instances.

Theorem 15. *Given a signal s, the best pure strategy of $\mathcal{D}$ in an SRG-v game on line graph, cycle graph, and SIMPLE–STAR graph instances can be found in polynomial time.*

Proof. For the sake of clarity, we describe the algorithm in the simplified case in which there is only one signal s. The extension to the general case is straightforward. The algorithm works as follows:

1. apply the algorithm searching for the route covering all the targets,
2. if the maximal covering route exists, then return it,
3. else remove the target t with the smallest $\pi(t)$ from $T(s)$,
4. go to Point 1.

Essentially, the algorithm returns the subset of targets admitting a covering route minimizing the maximum value among all the non-covered targets. □

Notice that the above results can be easily extended to tree graphs where the number of leaves is fixed.

Appendix C. Additional experimental results

We report in Fig. C.20 the boxplots of the results depicted in Fig. 5. They show that the variance of the compute times drastically reduces as ϵ increases. This is because the number of edges increases as ϵ increases and so the number of covering sets increases approaching $2^{|T|}$. On the other hand, with small values of ϵ, the number of covering sets of different instances can be extremely different.

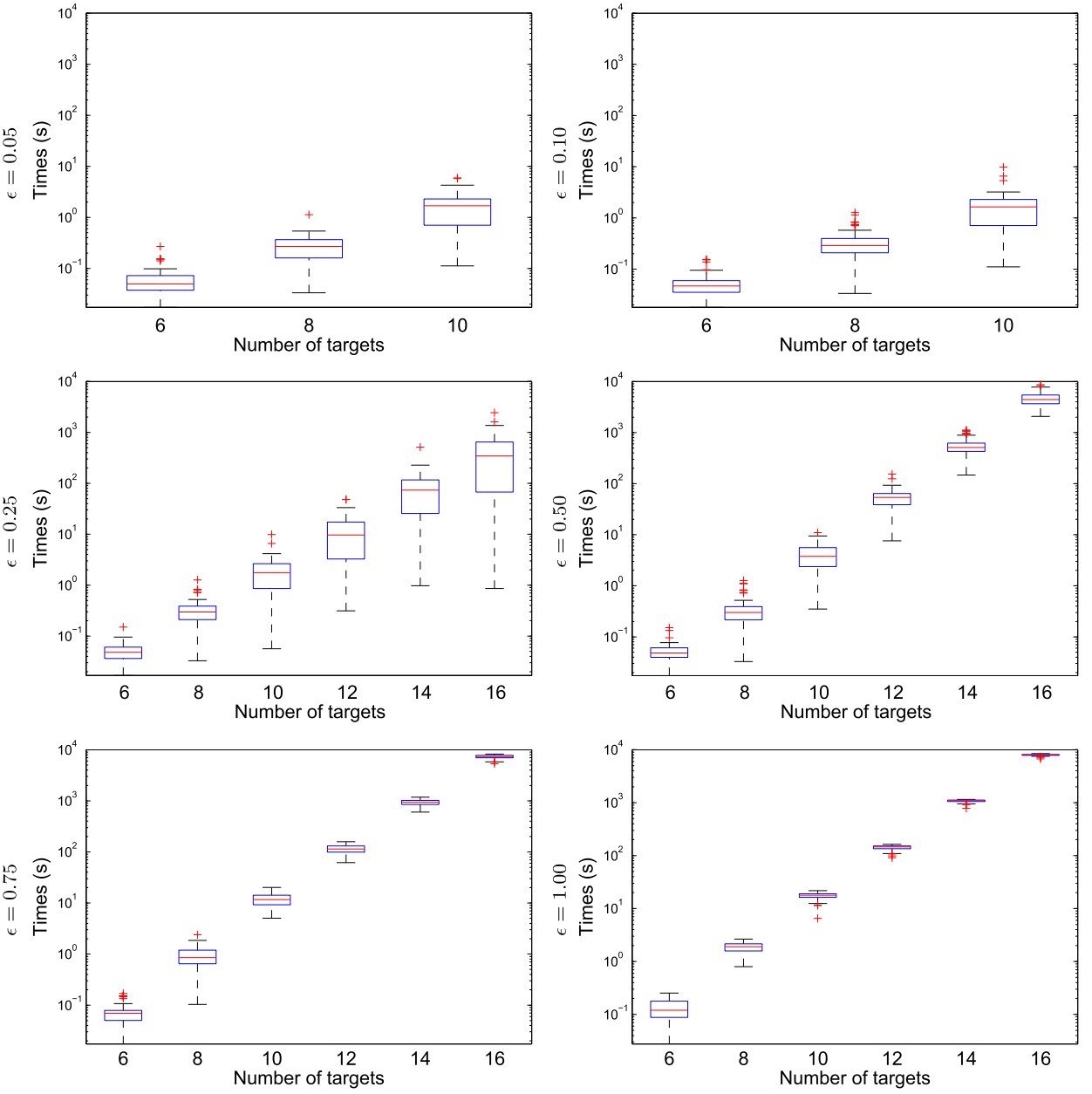


Fig. C.20. Boxplots of compute times required by our exact dynamic programming algorithm.

Appendix D. Notation

We report in Table D.3 the symbols used along the paper.

Table D.3
Symbols table.

	Symbol	Meaning
Basic model	$\mathcal{A}$	Attacker
	$\mathcal{D}$	Defender
	G	Graph
	V	Set of vertices
	v	Vertex
	v_i	i-th vertex
	E	Set of edges
	(v, v')	Edge
	$\omega^*_{v,v'}$	Temporal cost (in turns) of the shortest path between v and v'
	T	Set of targets
	t	Target
	t_i	i-th target
	$\pi(t)$	Value of target t
	$d(t)$	Penetration time of target t
Signals	S	Set of signals
	s	Signal
	p	Function specifying the probability of having the system generating signal s given that target t has been attacked
	$T(s)$	Targets having a positive probability of raising s if attacked
	$S(t)$	Signals having a positive probability of being raised if t is attacked
	$\perp$	No signals have been generated
	$\triangle$	No targets are under attack
Routes	$R_{v,s}$	Set of routes starting from vertex v when signal s is generated
	r	Route
	r_i	i-th route
	$r(i)$	i-th element visited along route r
	$U_{\mathcal{A}}(r_i, t_i)$	Attacker's utility given a route r and a target t
	$\sigma^{\mathcal{D}}$	Defender's strategy
	$\sigma^{\mathcal{D}}_v$	Defender's strategy starting from vertex v
	$\sigma^{\mathcal{D}}_{v,s}$	Defender's strategy starting from vertex v when signal s is generated
	$\sigma^{\mathcal{A}}$	Attacker's strategy
	$\sigma^{\mathcal{A}}_v$	Attacker's strategy when $\mathcal{D}$ is in v
	g_v	Value of the game (utility of $\mathcal{A}$)
	$A(r(i))$	Time needed by $\mathcal{D}$ to visit $r(i)$ starting from $r(0)$
	$T(r)$	Set of targets covered by route r
	$c(r)$	Temporal cost (in turns) associated to r

References

[1] M. Jain, B. An, M. Tambe, An overview of recent application trends at the AAMAS conference: security, sustainability, and safety, AI Mag. 33 (3) (2012) 14–28.
[2] B. Von Stengel, S. Zamir, Leadership with commitment to mixed strategies, Tech. rep., 2004.
[3] V. Conitzer, T. Sandholm, Computing the optimal strategy to commit to, in: Proceedings of the ACM Conference on Electronic Commerce, ACM EC, 2006, pp. 82–90.
[4] J. Pita, M. Jain, C. Western, C. Portway, M. Tambe, F. Ordóñez, S. Kraus, P. Paruchuri, Deployed ARMOR protection: the application of a game-theoretic model for security at the Los Angeles International Airport, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2008, pp. 125–132.
[5] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, M. Tambe, IRIS – A tool for strategic security allocation in transportation networks, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2009, pp. 1327–1334.
[6] B. An, E. Shieh, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, G. Meyer, Protect – A deployed game theoretic system for strategic security allocation for the United States coast guard, AI Mag. 33 (4) (2014) 96–110.
[7] F.M. Delle Fave, A.X. Jiang, Z. Yin, C. Zhang, M. Tambe, S. Kraus, J. Sullivan, Game–theoretic security patrolling with dynamic execution uncertainty and a case study on a real transit system, J. Artif. Intell. Res. 50 (2014) 321–367.
[8] B. Ford, D. Kar, F.M. Delle Fave, R. Yang, M. Tambe, PAWS: adaptive game-theoretic patrolling for wildlife protection, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2014, pp. 1641–1642.
[9] N. Basilico, N. Gatti, F. Amigoni, Patrolling security games: definition and algorithms for solving large instances with single patroller and single intruder, Artif. Intell. 184–185 (2012) 78–123.
[10] N. Basilico, N. Gatti, F. Villa, Asynchronous multi-robot patrolling against intrusion in arbitrary topologies, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2010, pp. 1224–1229.
[11] N. Agmon, G.A. Kaminka, S. Kraus, Multi-robot adversarial patrolling: facing a full-knowledge opponent, J. Artif. Intell. Res. 42 (2011) 887–916.
[12] E. Sless, N. Agmon, S. Kraus, Multi-robot adversarial patrolling: facing coordinated attacks, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2014, pp. 1093–1100.
[13] N. Agmon, C. Fok, Y. Emaliah, P. Stone, C. Julien, S. Vishwanath, On coordination in practical multi-robot patrol, in: Proceedings of the IEEE International Conference on Robotics and Automation, ICRA, 2012, pp. 650–656.
[14] Y. Vorobeychik, B. An, M. Tambe, S.P. Singh, Computing solutions in infinite-horizon discounted adversarial patrolling games, in: Proceedings of the International Conference on Automated Planning and Scheduling, ICAPS, 2014, pp. 314–322.
[15] E.A. Shieh, M. Jain, A.X. Jiang, M. Tambe, Efficiently solving joint activity based security games, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 2013, pp. 346–352.

[16] J. Gan, B. An, Y. Vorobeychik, Security games with protection externalities, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2015, pp. 914–920.
[17] N. Agmon, On events in multi-robot patrol in adversarial environments, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2010, pp. 591–598.
[18] S. Alpern, A. Morton, K. Papadaki, Patrolling games, Oper. Res. 59 (5) (2011) 1246–1257.
[19] N. Basilico, S. Carpin, T. Chung, Distributed online patrolling with multi-agent teams of sentinels and searchers, in: Proceedings of the International Symposium on Distributed Autonomous Robotic Systems, DARS, 2014, pp. 3–16.
[20] B.C. Ko, J.O. Park, J.-Y. Nam, Spatiotemporal bag-of-features for early wildfire smoke detection, Image Vis. Comput. 31 (10) (2013) 786–795.
[21] A.-J. Garcia-Sanchez, F. Garcia-Sanchez, J. Garcia-Haro, Wireless sensor network deployment for integrating video-surveillance and data-monitoring in precision agriculture over distributed crops, Comput. Electron. Agric. 75 (2) (2011) 288–303.
[22] J. Yick, B. Mukherjee, D. Ghosal, Wireless sensor network survey, Comput. Netw. 52 (12) (2008) 2292–2330.
[23] Z. Sun, P. Wang, M.C. Vuran, M.A. Al-rodhaan, A.M. Al-dhelaan, I.F. Akyildiz, Bordersense: border patrol through advanced wireless sensor networks, Ad Hoc Netw. 9 (3) (2011) 468–477.
[24] A. Krause, A. Roper, D. Golovin, Randomized sensing in adversarial environments, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 2011, pp. 2133–2139.
[25] E. Munoz de Cote, R. Stranders, N. Basilico, N. Gatti, N. Jennings, Introducing alarms in adversarial patrolling games, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2013, pp. 1275–1276.
[26] N. Basilico, N. Gatti, Strategic guard placement for optimal response to alarms in security games, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2014, pp. 1481–1482.
[27] N. Basilico, G. De Nittis, N. Gatti, Multi-resource defensive strategies for patrolling games with alarm systems, CoRR, arXiv:1606.02221, http://arxiv.org/abs/1606.02221.
[28] N. Basilico, G. De Nittis, N. Gatti, A security game combining patrolling and alarm-triggered responses under spatial and detection uncertainties, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2016, pp. 397–403.
[29] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, F. Amigoni, Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings, in: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT, 2009, pp. 557–564.
[30] N. Basilico, N. Gatti, T. Rossi, Capturing augmented sensing capabilities and intrusion delay in patrolling-intrusion games, in: Proceedings of the IEEE Symposium on Computational Intelligence and Games, CIG, 2009, pp. 186–193.
[31] Y. Shoham, K. Leyton-Brown, Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations, Cambridge University Press, New York, NY, USA, 2008.
[32] M. Maschler, S. Zamir, E. Solan, Game Theory, Cambridge University Press, 2013.
[33] L.S. Shapley, R.N. Snow, Basic solutions of discrete games, Ann. Math. Stud. 24 (1950) 27–35.
[34] M.J. Osborne, An Introduction to Game Theory, vol. 3, Oxford University Press, New York, 2004.
[35] H.-J. Böckenhauer, J. Hromkovic, J. Kneis, J. Kupke, The parameterized approximability of tsp with deadlines, Theory Comput. Syst. 41 (3) (2007) 431–444.
[36] N. Bansal, A. Blum, S. Chawla, A. Meyerson, Approximation algorithms for deadline-tsp and vehicle routing with time-windows, in: Proceedings of the Annual ACM Symposium on Theory of Computing, STOC, 2004, pp. 166–174.
[37] M.W. Savelsbergh, Local search in routing problems with time windows, Ann. Oper. Res. 4 (1) (1985) 285–305.
[38] H.-J. Böckenhauer, L. Forlizzi, J. Hromkovič, J. Kneis, J. Kupke, G. Proietti, P. Widmayer, Reusing optimal tsp solutions for locally modified input instances, in: Proceedings of the IFIP International Conference on Theoretical Computer Science, TCS, 2006, pp. 251–270.
[39] D. Korzhyk, Z. Yin, C. Kiekintveld, V. Conitzer, M. Tambe, Stackelberg vs. Nash in security games: an extended investigation of interchangeability, equivalence, and uniqueness, J. Artif. Intell. Res. 41 (2011) 297–327.
[40] B. An, M. Tambe, F. Ordóñez, E.A. Shieh, C. Kiekintveld, Refinement of strong Stackelberg equilibria in security games, in: Proceedings of the Conference on Artificial Intelligence, AAAI, 2011, pp. 587–593.
[41] M. Jain, C. Kiekintveld, M. Tambe, Quality–bounded solutions for finite Bayesian Stackelberg games: scaling up, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2011, pp. 997–1004.
[42] M. Brown, B. An, C. Kiekintveld, F. Ordóñez, M. Tambe, An extended study on multi-objective security games, Auton. Agents Multi-Agent Syst. 28 (1) (2014) 31–71.
[43] T.H. Nguyen, R. Yang, A. Azaria, S. Kraus, M. Tambe, Analyzing the effectiveness of adversary modeling in security games, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2013, pp. 718–724.
[44] B. An, M. Brown, Y. Vorobeychik, M. Tambe, Security games with surveillance cost and optimal timing of attack execution, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2013, pp. 223–230.
[45] R. Yang, B. Ford, M. Tambe, A. Lemieux, Adaptive resource allocation for wildlife protection against illegal poachers, in: Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS, 2014, pp. 453–460.
[46] B. An, F. Ordóñez, M. Tambe, E. Shieh, R. Yang, C. Baldwin, J. DiRenzo, K. Moretti, B. Maule, G. Meyer, A deployed quantal response-based patrol planning system for the U.S. Coast Guard, Interfaces 43 (5) (2013) 400–420.
[47] R. Yang, A.X. Jiang, M. Tambe, F. Ordóñez, Scaling-up security games with boundedly rational adversaries: a cutting-plane approach, in: Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI, 2013, pp. 404–410.
[48] J. Tsai, T.H. Nguyen, N. Weller, M. Tambe, Game-theoretic target selection in contagion-based domains, Comput. J. 57 (6) (2014) 893–905.
[49] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, M. Tambe, Computing optimal randomized resource allocations for massive security games, in: Proceedings of the International Joint Conference on Autonomous Agents and Multi-Agent Systems, AAMAS, 2009, pp. 689–696.
[50] A. Blum, N. Haghtalab, A.D. Procaccia, Lazy defenders are almost optimal against diligent attackers, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI, 2014, pp. 573–579.
[51] C.H. Papadimitriou, M. Yannakakis, The traveling salesman problem with distances one and two, Math. Oper. Res. 18 (1) (1993) 1–11.
[52] R.W. Conway, W.L. Maxwell, L.W. Millerr, Theory of Scheduling, Dover Books on Mathematics, 2003.