Balancing Unpredictability and Coverage in Adversarial Patrolling Settings

Nicola Basilico[1] and Stefano Carpin[2]

[1] Department of Computer Science
University of Milan, Italy
[2] Department of Computer Science and Engineering
University of California, Merced, CA, USA

Abstract. We present a novel strategy for a patroller defending a set of heterogeneous assets from the attacks carried by an attacker that through repeated observations attempts to learn the strategy followed by the patroller. Implemented through a Markov chain whose stationary distribution is a function of the values of the assets being defended and the topology of the environment, the strategy is biased towards providing more protection to valuable assets, yet is provably hard to learn for an opponent. After having studied its properties, we show that our proposed method outperforms strategies commonly used for this type of problems.

Keywords: Security Games; Learning; Patrolling.

1 Introduction

Intelligent autonomous systems are increasingly utilized for providing surveillance to valuable or sensitive assets, such as harbors, banks, airports and the like [18]. Such systems may be implemented by deploying robots in the environment, or as decision support systems informing one or more human operators about how to complete the task. In the following, we use the terms *agents* or *patrollers* to indicate the entities protecting the assets. In non trivial scenarios assets are spatially distributed, and when the number of assets to guard is larger than the number of agents used to protect them, patrollers need to move between different assets to ensure coverage. Central to this problem, then, is a scheduling algorithm deciding the route an agent should follow. In competitive situations where patrollers face adversarial entities trying to compromise the assets being protected, deterministic patrolling routes are normally avoided because through repeated observations an intelligent adversary could easily determine the deterministic route, and then attack an asset while being sure it is not protected. Randomization plays then an essential role in these problems. However, purely random strategies are not ideal either, in particular if the assets are heterogeneous and of different value. In such case the patroller should devote more effort to defend the more valuable assets, i.e., introduce some bias in the route towards the more valued locations. This approach, however, is also prone to being learned and forecast by an intelligent opponent that can build a model of the process used to generate patrolling routes.

In this paper, we present a novel patrolling strategy for a single patroller that is provably hard to model or learn for an adversarial opponent. Notwithstanding, the strategy is not purely random, but rather biased to provide more coverage to the most important assets. Central to our approach is the idea of allowing the patroller to *slow down* when traveling between vertices, thus introducing unpredictability in the time it takes to move from asset to asset. As we will show, an opponent trying to determine how often a patroller visits a certain region will essentially be faced with the problem of forecasting a white noise time series. To ensure that more valuable assets are visited more often, the next vertex to visit is selected using a Markov chain whose stationary distribution considers the structure of the underlying graph and ensures that expected losses are bounded in a worst case scenario. The original contributions of this paper are the following:

– we consider an adversarial patrolling setting where we assume realistic observation capabilities for the attacker;
– we devise an approach to approximate a strategy that is optimal against a fully informed attacker and that makes information gathering through observation a difficult process, independently on how the attacker formulates its beliefs;
– we empirically evaluate the proposed approach in realistic settings and we compare it with a mainstream approach for robotics surveillance applications.

The rest of this paper is organized as follows. Related work is discussed in Section 2, where we also outline how our work differs, while the adversarial patrolling problem we consider is formally introduced in Section 3. Next, in Section 4 we recap a few facts about Markov Chains necessary to prove the properties of the patrolling algorithm we then present and study in Section 5. Building upon our theoretical findings, in Section 6 we show how our algorithm defeats commonly used alternatives. Finally, conclusions are given in Section 7.

2 Related Work

The use of mobile robots in surveillance settings received considerable attention from the Robotics and AI communities in the recent years. The classic problem formulation, which is also the one we adopt in this work, describes the environment by means of a graph whose vertices represent areas to protect. To safeguard these assets, one or more patrolling agents travel on this graph, trying to detect and stop ongoing intrusions in each currently visited vertex.

A *patrolling strategy* is used to determine the movements of the patrolling agents. Its general objective is to schedule the patrolling activities to optimize the enforced protection in the environment and the execution costs. A first theoretical analysis that recognized the importance of this problem was presented in [8]. Therein, the author studies the optimality of particular classes of patrolling strategies adopting a criterion based on the *idleness of a vertex*, defined as the time elapsed since its last visit by a patroller. Idleness-based criteria naturally encode the intuitive rationale that the more frequently a vertex is patrolled, the more protected it will be. This type of approaches are today among the mainstream solutions for a large number of applications deployed in the real world [13]. Most of the idleness-based settings entail hard optimization problems, making the study of heuristic and approximated solutions an important goal. For example, the work proposed in [2] studies a setting where idleness is combined with the importance of the different vertices and provides approximation algorithms able to scale up to very large instances.

A parallel line of research focuses on stochastic surveillance strategies based on Markov chains and Monte Carlo techniques. The general idea in these works is to devise strategies that could exhibit desired properties at convergence. For example, in [9] the authors define the patrolling strategy as a Markov chain and seek the minimization of the mixing rate to a desired steady state uniform distribution for an homogeneous setting where all vertices have equal importance. A more recent example is presented in [1], where the objective to be minimized is the mean first passage time.

All the above approaches, either deterministic or stochastic, do not consider the adversarial nature of the patrolling task. Along such dimension, they are complemented by the wide literature on security games [16], that addresses the general problem of resource allocation in the presence of rational adversaries and that includes models specifically tailored for patrolling on graphs [3, 5]. The working assumption of these approaches is that the patrolling task is carried out in the presence of a rational attacker that, thanks to unlimited observation capabilities, has exact knowledge of the patrolling strategy. This assumption results in a leader-follower interaction where the attacker substantially best responds to the patrolling strategy it observes.

Some model refinements considered security games where the attacker has limited or constrained observation capabilities. One notable example is [4], where the attacker constructs a belief over the patrolling strategy by performing costly observations. In [6] the authors consider a similar belief-based observing attacker and show that planning against the strongest observer induces limited losses.

In a departure from former literature in this area, in this work we consider a patrolling setting characterized by an observing attacker and we try to overcome the predictability of the widely-adopted deterministic idleness-based patrolling strategies. To this end we seek a stochastic approach based on Markov chains that, instead of focusing on a particular convergence property, seeks a steady-state distribution that complies with the area coverage requirements while, at the same time, being difficult to predict for an observer. Differently from the classical security games literature, we do not assume a setting where the attacker has exact knowledge of the patrolling strategy that is being executed. With respect to security games that considered refinements of the observer, we adopt a much more limited (and realistic) attacker model, i.e., one that cannot obtain full knowledge of the patrolling strategy even with unbounded time or observations. We maintain that such a model is more suitable to capture realistic interactions between a robotic patrolling system and a malicious attacker in a physical environment represeted by a graph.

3 Problem Definition

Our patrolling problem is defined as follows. We consider a domain $\mathcal{D}$ within which K target locations are found. Let the target locations be $l_1, \ldots, l_K$. An undirected, weighted graph $G = (V, E)$ is used to model this environment. Each target location is associated with a vertex in G, and the weight of and edge (l_i, l_j), denoted as $d_{i,j} \in \mathbb{R}_+$, represents the temporal traveling cost when moving from l_i to l_j (or viceversa). We assume the graph is connected and complete[3] and we furthermore set $d_{i,i} = 0$ for all vertices, even though self-loop edges do not necessarily need to be assumed. Edge costs are assumed to be symmetric, and therefore $d_{i,j} = d_{j,i}$ Each target is associated with a *value* v_i, which encodes a measure of its importance, and a *time to attack* a_i, which will be explained later.

The task we consider is that of controlling one patrolling agent moving over the graph G with the objective of protecting its targets. Differently from what typically done in graph exploration settings, we do not assume non-preemptive edge traversals. The only constraint we enforce is that when leaving a vertex l_i the patroller cannot arrive at another vertex l_j before at least $d_{i,j}$ time units have passed. Moreover, the patroller is assumed to have attack detection capabilities localized to the currently occupied vertex. We assume to perform this task under a threat modeled as an *attacker* agent whose objective is compromising one of the target locations. To successfully carry out an attack on target l_i, the attacker has to position itself at l_i for a time greater or equal than the attack time a_i. During such time, if the patroller visits l_i before the attack time expires, the attacker is neutralized and the attack fails. Figure 1 illustrates this interaction, where, as customary in literature, we assume that payoffs reflect a constant-sum setting. Specifically, defining $\nu = \sum_{l_i \in V} v_i$, we assume that when the attacker successfully carries out an attack to target l_i it receives a payoff of v_i while the patroller gets $\nu - v_i$. If instead the attack is detected the attacker receives 0 while the patroller gets ν.

The patroller's sequence of visits to the targets is defined by a patrolling strategy that is not accessible to the attacker. However, the attacker can observe a specific single target l_i and, at any time, it can determine if the patroller is currently at that vertex or not. Formally, this means that the attacker is capable of recording the timestamps of the sequence of visits that the patroller performs at target l_i and, from such list of timestamps, it can compute the inter-arrival times at l_i. This sequence represents the only source of knowledge available to the attacker. Based on this setup, the following two facts easily follow:

[3]The assumption is w.l.o.g. since any non-complete graph can be turned into a complete one by taking its closure on shortest paths, i.e., adding an edge (l_i, l_j) and setting $d_{i,j}$ to the cost of the shortest path between l_i and l_j in the original graph.

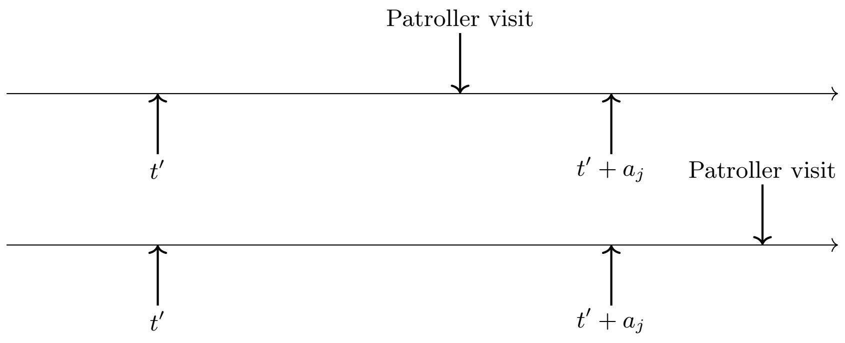

Fig. 1. Two scenarios where the attacker attacks location l_j at time t'. In the first case (top) the next visit to l_j by the patroller happens before $t' + a_j$, the attacker is detected and the attack fails. In the second case (bottom) the next visit happens after $t' + a_j$ and the attacker is therefore successful and receives reward v_j.

- the attacker will attack the observed vertex, but the patroller does not know the vertex observed by the attacker;
- the patroller cannot increase the chances of detecting an attack by spending time at a vertex since a rational attacker will never strike a target that is under surveillance; in practice, as soon as it visits a vertex, the patroller leaves it.

The question we aim at answering is therefore the following. How should the patroller schedule its visits to the locations, so that the expected reward gathered by the attacker is minimized? Notice that, for now, we formulate our question without making any assumption on the algorithm used by the attacker to translate the gathered information into an attack decision.

4 Background in Markov Chains

We shortly recap some basic facts about Markov chains that will be useful to establish the properties of the algorithm we propose in the following section. For sake of brevity, the discussion remains informal. For more details and a rigorous discussion, the reader is referred to textbooks in stochastic processes, such as [11, 12].

A Markov chain with finite state space is defined by a finite set of states $\{s_1, \ldots, s_n\}$ and an $n \times n$ transition matrix $\boldsymbol{P}$ whose (i, j)-th entry $p_{i,j}$ defines the one-step transition probability, i.e., $p_{i,j} = \Pr[X_{t+1} = s_j | X_t = s_i]$, where X_t is the state of the Markov chain at time t. The matrix $\boldsymbol{P}$ is stochastic, i.e., its rows add to one. A Markov chain is *irreducible* if from each state it is possible to move to any other state in one or more steps. A state s in a Markov chain is *periodic* if starting from s the chain can return to s only after a number of steps multiple of an integer larger than one. A state is *aperiodic* if it is not periodic. In an irreducible Markov chain, if one state is aperiodic, then all states are aperiodic. Based on these definitions, it is immediate to observe that if all entries in $\boldsymbol{P}$ are strictly positive, then the Markov chain is irreducible and aperiodic. If a Markov chain is irreducible and aperiodic, there always exist a unique stationary distribution, i.e., an n-dimensional stochastic vector $\boldsymbol{\pi}$ such that

$$\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}.$$

The stationary distribution $\boldsymbol{\pi}$ describes the probability that, irrespective of the initial state, the Markov chain is in a given state once it has converged, i.e., π_i is the probability that the Markov chain is in s_i after a large number of transitions. For a given Markov chain, the *recurrent visit time* for a state s_i is the number of transitions until the chain returns to state s_i given that it started in state s_i. The recurrent visit time for a vertex s_i is a discrete random variable whose expectation is $\frac{1}{\pi_i}$ where π_i is the i-th component of the stationary distribution $\boldsymbol{\pi}$.

4.1 Relationships Between $\boldsymbol{\pi}$ and $\boldsymbol{P}$

As stated above, if a Markov chain is irreducible and aperiodic, then the stationary distribution exists and is unique. This means that $\boldsymbol{\pi}$ is a function of $\boldsymbol{P}$. Often times we are interested in the opposite problem, i.e., for a given stationary distribution $\boldsymbol{\pi}$ we want a transition matrix $\boldsymbol{P}$ having $\boldsymbol{\pi}$ as stationary distribution. In general there exist multiple transition matrices solving this problem. The Metropolis-Hastings algorithm can be used to determine one such matrix. In simplified form[4] the method to determine $\boldsymbol{P}$ can be formulated as follows. For $i \neq j$ define

$$\alpha_{i,j} = \min\left\{1, \frac{\pi_j}{\pi_i}\right\} \qquad p_{i,j} = \frac{1}{n}\alpha_{i,j}$$

For $i = j$, p_{ii} is instead set so that all rows add up to one. Therefore, for a given $\boldsymbol{\pi}$ this method provides a unique transition matrix $\boldsymbol{P}$, although this is not the only matrix such that $\boldsymbol{\pi} = \boldsymbol{\pi}\boldsymbol{P}$.

5 An Unforecastable Patrolling Strategy

In this section we develop a randomized patrolling strategy aimed at being "difficult" to observe for the attacker we introduced in Section 3, and, at the same time, complying with the environment coverage requirements (visiting more frequently the vertices with higher values) and biased to bound its worst case losses. To ease the discussion of the final strategy, we start with a generic approach that will then be instantiated to an algorithm with an accurate performance characterization. The framework is schematized in Algorithm 1, where we outline

[4]This simplified version is obtained assuming that the proposal distribution is $\frac{1}{n}$ for all i, j. See [14] for more details.

a generic patrolling strategy whose core idea is that of decoupling the spatial decision from the temporal one.

```
1 Randomly select a start vertex $l_i \in V$;
2 while true do
3     Randomly select the next vertex $l_j \in V$;
4     Generate a random time $t \geq d_{i,j}$ with finite expectation;
5     Move to $l_j$ spending time $t$;
6     $l_i \leftarrow l_j$;
```

Algorithm 1: General patrolling strategy

The patroller operates in an infinite loop where in line 3 it selects the next vertex to visit, and in line 4 it determines how much time to spend to move there. The random variables describing these choices are subject to the constraints discussed in the following. First, when selecting the new vertex l_j, a strictly positive probability must be assigned to each vertex in V. This does not necessarily imply a uniform distribution, and indeed in the final formulation a uniform distribution will not be used. We assume these probabilities are however constant and therefore history independent. Therefore, they are not influenced by the idleness of a vertex, i.e., the time since its last visit. Second, when moving towards the selected vertex l_j, the patroller does not necessarily spend the minimum time $d_{i,j}$, but rather elects to spend a time t that is larger or equal than $d_{i,j}$. This is modeled by a continuous random variable with zero density for values smaller than $d_{i,j}$ and finite expectation.[5] This ability to *slow down* is essential to introduce unpredictability in the patrolling schedule and therefore make it difficult for an observer to reconstruct the sequence of inter-arrival times to a particular target. In particular, it is critical to observe that if in step 3 the selected next vertex l_j is equal to the current vertex l_i, the time spent to *move* from l_i to l_i can be strictly positive even though $d_{i,i} = 0$. This, for example, corresponds to the situation where the patroller first leaves l_i, but then comes back to l_i again before having visited another vertex.

Remark: the importance of this last aspect should not be overlooked. Consider the classic case where from l_i the patroller would always pick a next vertex different from l_i (or, if deciding to remain in l_i, it would not leave it, i.e., it would move from l_i to l_i with time $d_{i,i} = 0$.) Through repeated observations, an opponent could easily figure out that when the patroller leaves l_i, the next visit will happen no earlier than $m_i = 2\min_{j \neq i} d_{i,j}$ (this is the time to move forward and backward along the outgoing edge of minimum cost). Based on this information, the attacker could strategically determine that if $a_i < m_i$ then an attack to l_i placed immediately after the patroller leaves l_i would succeed with probability 1. Instead, with the proposed patrolling strategy the time for the next visit to l_i could be smaller than m_i and therefore the attacker cannot produce an attack

[5]The density must be 0 for values smaller than $d_{i,j}$ because the patroller must generate a time larger than the minimum time necessary to complete the move.

that is certain to succeed.

If we next consider the temporal sequence of vertices visited by the patroller and the assumptions we made about how the next vertex is selected, this can be modeled as a finite Markov chain whose state space is V. Let $\boldsymbol{P}$ be the associated $K \times K$ transition matrix. Because of the assumptions we made about how the next vertex is selected in line 3, it follows that all entries in $\boldsymbol{P}$ are strictly positive and therefore the Markov chain is irreducible and aperiodic. Let us now consider the time elapsed between two successive visits to vertex l_j by the patroller. As shown in Figure 2, this is a continuous random variable R_j obtained adding a finite number of continuous random variables. In particular, each of these is generated in step 4 in Algorithm 1. This continuous random variable is in the following called *return time to vertex* l_j to distinguish it from the discrete recurrent visit time introduced in Section 4.

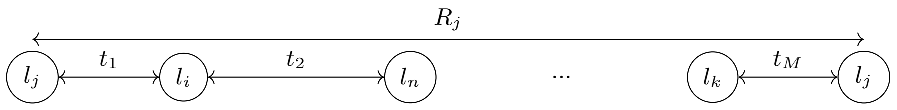

Fig. 2. The return time to vertex l_j (R_j) is the overall time spent by the patroller to return in l_j given that it started in l_j. R_j is the sum of M random variables $t_1 \dots t_M$, where M is the recurrent visit time for vertex l_j.

To gain some insights about the random variable R_j, it is necessary to now make some assumptions about how the random times are generated in line 4 in Algorithm 1. To this end, we assume that the time is distributed as a uniform random variable between the values $d_{i,j}$ and $d_{i,j} + \Delta$, where $\Delta > 0$ is a positive constant. In the following, with a slight abuse of notation we indicate as $\mathcal{U}[d_{i,j}, d_{i,j} + \Delta]$ both this random variable and its density. This choice respects the constraints we assumed earlier on, i.e., its density is 0 for values smaller than $d_{i,j}$ and it has finite expectation. Let us now consider the sequence of travel times generated by the patroller while moving from vertex to vertex. Because of the assumption we made about how the travel time is generated when moving between l_i and l_j, and since the Markov chain is irreducible and aperiodic, it follows that travel times are distributed according to a mixture of random variables with the following density:

$$\sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} p_{i,j} \mathcal{U}[d_{i,j}, d_{i,j} + \Delta] \tag{1}$$

where π_i is the ith entry in the unique stationary distribution $\boldsymbol{\pi}$ associated with $\boldsymbol{P}$. This random variable has finite expectation equal to

$$\sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} p_{i,j} . \frac{2d_{i,j} + \Delta}{2} \tag{2}$$

Importantly, all travel times are independent, identically distributed (iid) according to this mixture of uniforms. This allows to determine the expectation of R_j for each location l_j, as detailed by the following theorem.

Theorem 1. *Let us consider the Markov Chain induced by Algorithm 1 and let R_j be the return time to state l_j. Its expectation is*

$$\mathbb{E}[R_j] = \frac{1}{\pi_j} \sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} p_{i,j} \frac{2d_{i,j} + \Delta}{2}$$

where $\boldsymbol{\pi}$ is the stationary distribution and $p_{i,j}$ is the (i, j) entry in the transition matrix $\boldsymbol{P}$.

Proof. First recall that since all entries in $\boldsymbol{P}$ are strictly positive, the Markov chain is recurrent and aperiodic, and therefore there exist a unique stationary distribution $\boldsymbol{\pi}$. With reference to Figure 2, R_j can be written as

$$R_j = \sum_{i=1}^{M_j} t_i$$

where each of the t_i is a random variable with density given by Eq. (1) and M_j is the random variable for the recurrence time for state l_j. R_j is therefore the sum of a random finite number of iid random variables. The claim, therefore, follows by invoking Wald's equation (see, e.g., [14], Theorem 7.2).

$$\mathbb{E}[R_j] = \mathbb{E}\left[\sum_{i=1}^{M_j} t_i\right] = \mathbb{E}[M_j]\mathbb{E}[t_i] = \frac{1}{\pi_j} \sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} p_{i,j} \frac{2d_{i,j} + \Delta}{2}$$

where we used the aforementioned fact that $\mathbb{E}[M_j] = \frac{1}{\pi_j}$ and Eq. (2) for $\mathbb{E}[t_i]$.

Remark: the attentive reader could correctly object that π_i is the probability that the Markov chain is in state l_i only after the chain has converged, and therefore the t_is will be iid only in the limit. However, as we we will show later on, the algorithm will bootstrap its operations ensuring that that the Markov chain has already converged from the first step, and therefore the t_is are always iid as required in the proof.

To determine $\mathbb{E}[R_j]$ one needs to know $\boldsymbol{P}$ (from which $\boldsymbol{\pi}$ can be derived). $\boldsymbol{P}$ is determined by the the patroller selects the next vertex in algorithm 1, line 3. Alternatively, we could determine a desired stationary distribution $\boldsymbol{\pi}$ and from this reconstruct a transition matrix $\boldsymbol{P}$ using the Metropolis-Hastings method we discussed in Section 4.1. Such matrix could then be used to implement the selection of the next vertex to visit. In the following section we show that indeed the transition matrix will be computed starting from a target stationary distribution.

5.1 Optimizing Against an Observing Attacker

As customary done in the security games literature, we take a worst-case stance and assume that the attacker is in the position of observing a vertex l_j for an indefinite amount of time. This, however, does not amount to assume that the attacker has knowledge of the patrolling strategy, but, instead, that the attacker can build a correct belief on the random variable R_j. The expected payoff obtained by attacking l_j immediately after the patroller leaves l_j is then

$$v_j \Pr[R_j > a_j],$$

where $\Pr[R_j > a_j]$ is the probability that the attack will be successful since $R_j > a_j$ corresponds to the event in which the patroller returns on l_j only after the attack time a_j has elapsed (recall Figures 1 and 2). This probability is in general difficult to compute analytically, although it could be approximated numerically. However, being R_j a non-negative random variable, we can exploit Markov's inequality [10] to bound the probability of a successful attack:

$$\Pr[R_j \geq a_j] \leq \frac{\mathbb{E}[R_j]}{a_j}$$

It follows that the attacker's expected payoff obtained on l_j is upper bounded by

$$L_j = \frac{v_j}{a_j} \frac{1}{\pi_j} \sum_{i=1}^{K} \pi_i \sum_{j=1}^{K} p_{i,j} \zeta_{i,j} \tag{3}$$

where we used Theorem 1 for $\mathbb{E}[R_j]$ and we wrote $\zeta_{i,j}$ for $\frac{2d_{i,j}+\Delta}{2}$. Note that $\zeta_{i,j}$ is a function of the graph and Δ only, but does not depend on the strategy.

Since we assumed to work in a constant-sum setting, the expression for L_j defines an upper bound on the patroller's expected loss that we seek to minimize. Specifically, let $\mathcal{S}$ be the set of all possible strategies obtained by varying the vertex selection step in algorithm 1. Under the operational assumption that we do not know in advance the target observed by the attacker, we define the patrolling strategy as the solution of the following optimization problem:

$$\min_{s \in \mathcal{S}} \max_{j=1 \ldots K} L_j$$

This can be rewritten as follows:

$$\begin{aligned} & \min_{\boldsymbol{\pi}} \max_{j} \frac{v_j}{a_j} \frac{1}{\pi_j} \sum_{i=1}^{K} \pi_i \sum_{k=1}^{K} p_{i,k} \zeta_{i,k} \qquad j = 1 \ldots K \\ s.t. \quad & \pi_1 + \pi_2 + \cdots + \pi_K = 1 \\ & \pi_j > 0 \qquad j = 1 \ldots K \end{aligned} \tag{4}$$

where the optimization variable is the K dimensional stochastic vector $\boldsymbol{\pi}$ and we assume that the $p_{i,j}$ values are obtained from $\boldsymbol{\pi}$ using the method described in Section 4.1.

Theorem 2. *Assuming $\boldsymbol{P}$ is computed from $\boldsymbol{\pi}$ using the Metropolis-Hastings method given in section 4.1, the solution to the optimization problem* (4) *is obtained by setting*

$$\pi_i = \frac{\frac{v_i}{a_i}}{\sum_{j=1}^{K} \frac{v_j}{a_j}} \qquad 1 \leq i \leq K. \tag{5}$$

Proof. Let $\boldsymbol{\pi}^*$ be the vector obtained setting all components according to Eq. (5). The j-th function to be optimized in problem (4) is $\frac{v_j}{a_j}\mathbb{E}[R_j]$, where $\mathbb{E}[R_j]$ is the expected return time for vertex l_j induced by the strategy defined by $\boldsymbol{\pi}^*$. By substitution, it is immediate to verify that for this specific choice of $\boldsymbol{\pi}^*$ all j functions to be minimized assume the same value, i.e.,

$$\frac{v_1}{a_1}\mathbb{E}[R_1] = \frac{v_2}{a_2}\mathbb{E}[R_2] = \cdots = \frac{v_K}{a_K}\mathbb{E}[R_K]$$

By contradiction, assume $\boldsymbol{\pi}^*$ is not optimal. This means there exist a vector $\boldsymbol{\pi}' \neq \boldsymbol{\pi}^*$ giving a lower value for problem (4). Let $\mathbb{E}[R'_j]$ be the expected return time induced by $\boldsymbol{\pi}'$. For $\boldsymbol{\pi}'$ to be optimal, it must then be

$$\frac{v_j}{a_j}\mathbb{E}[R'_j] < \frac{v_j}{a_j}\mathbb{E}[R_j] \qquad j = 1 \ldots K$$

i.e., $\mathbb{E}[R'_j] < \mathbb{E}[R_j]$ for all vertices. However, this cannot be simultaneously achieved for all vertices, because if the expected return time for a vertex decreases, then there must be an increase in the return time for a different vertex.

At this point we have all elements to write the detailed patrolling strategy implemented by the patroller. The input is the graph $G = (V, E)$ together with the values and attack times for each vertex, and the parameter Δ.

```
1 Input: graph $G = (V, E)$, vector of values $\boldsymbol{v}$, vector of attack times $\boldsymbol{a}$,
  $\Delta$; Solve optimization problem (4) and determine $\boldsymbol{\pi}$;
2 Compute transition matrix $\boldsymbol{P}$ using Metropolis-Hastings method;
3 Select start vertex $l_i \sim \boldsymbol{\pi}$ ;
4 while true do
5     Select next vertex $l_j$ with probability $\boldsymbol{P}_{ij}$;
6     Generate random time $t \sim \mathcal{U}[d_{i,j}, d_{i,j} + \Delta]$;
7     Move to $l_j$ spending time $t$;
8     $l_i \leftarrow l_j$;
```

Algorithm 2: Adopted patrolling strategy

Remark: line in algorithm 2 uses the stationary distribution to select the initial vertex. This ensures that the Markov chain converges from the first step, i.e., $\boldsymbol{\pi}$ gives the probability of being in each state at each time step. This ensures that all travel times t_i are identically distributed according to the mixture of uniforms. If the start vertex was chosen differently, this would be true only in the limit, i.e., after the chain has converged.

Let us now consider what an attacker observing location l_j trying to construct a belief over R_j would “see”. The return times R_j are distributed as per Eq. (1). This entails that the attacker would observe a sequence iid, finite mean and variance inter-arrival times. It is useful to recall that independent variables with finite mean and variance are also uncorrelated and therefore the sequence of return times constitutes a white noise stochastic process. It is worth noting that some authors require zero mean in the definition of white noise, while some others do not and simply require uncorrelation between all values (see e.g., [11], page 385). In our case, the return times do not have zero mean, but this is inconsequential to the (in)ability of using past values to make predictions about future ones. The strategy summarized in Algorithm 2 is then consistent with the adversarial setting since it seeks optimality against a fully informed and rational attacker (as defined in Section 3). At the same time, however, it makes difficult to this type of attacker to reach such a fully informed condition by observing a target. It is also worth noting that the solution to problem (4) depends on the ratios v_j/a_j, i.e., the ratio between the value of a vertex and the time to attack. This quantity is as the amount of value per unit of time that an attacker will receive if the attack succeeds.

6 Experimental evaluation

In this section we provide some empirical assessments of the patrolling strategies obtained with Algorithm 2, which we will refer as “Delta”. As already discussed, “Delta” is the optimal strategy as per Problem 4. The experimental results we present here are focused on evaluating its advantages in terms of difficulty of being observed by an attacker characterized by the local-target visibility model we introduced in Section 3. We shall compare “Delta” with an heuristic strategy that well represents an established class of approaches for robotic surveillance (see our discussion in Section 2). We created a dataset of random graphs scaling up to 50 vertices. For each problem instance we execute the patrolling strategy in simulation for 5000 vertex visits. For the Delta strategy we set $\Delta = \max_{(i,j)} d_{i,j}/2$.

6.1 Comparing Against a Heuristic Strategy

We compare against a heuristic strategy based on vertex idleness. Formally, let $I(l_i)$ be the current idleness of vertex l_i defined as the amount of time since the last visit by the patroller. In the beginning, $I(l_i) = 0$ for all $l_i \in V$. To understand how $I()$ evolves during the patrolling mission, let us assume that at time t_1 the patroller leaves a vertex l_1, and it arrives at vertex l_2 at time $t_2 \geq t_1 + d_{1,2}$. Upon its arrival in l_2, the idleness for l_2 is set to $I(l_2) = 0$, while for $i \neq 2$ it is set to $I(l_i) = I(l_i) + (t_2 - t_1)$. Once in a vertex l_i, the patroller makes its decision on where to go next by computing the following utility function for each vertex $l_j \neq l_i$:

$$u(l_j) = \frac{v_j}{a_j}\left(1 + \frac{I(l_j)}{d_{i,j}}\right)$$

Maximizing the above utility would result in an informed choice, where important targets that have been left unvisited would be chosen. However, the resulting sequence of inter-arrival times that could be observed at a target would be easily forecastable since the strategy is ultimately deterministic and, in the long run, it will produce a periodic behavior. For this reason we add to the fully informed decision a random component, inspired by ε-soft exploration strategies [15]. The objective of the random component is to mislead an observer. We do this by introducing a random selection with a parameter $\varepsilon \in [0, 1]$ and define the selection of l^{next} (the next location to patrol) as follows (we assume that l_i is the current vertex occupied by the patroller):

$$l^{next} = \begin{cases} \arg\max_{l_j \in V \setminus \{l_i\}} u(l_j), \text{ with probability } 1 - \varepsilon \\ \text{choose uniformly in } V \setminus \{l_i\}, \text{ with probability } \varepsilon \end{cases}$$

One of the standard approaches to assess if a time series can be forecast from its past values is to perform a partial autocorrelation analysis [17]. Given a lag k, the partial autocorrelation function measures the likelihood that a sample observed k steps in the past can be a predictor for the last observed value of the series (the correlation between a sample and a previous one is computed assuming a linear relationship). Figure 3 shows such analysis for the inter-arrival times observed at particular target on a randomly generated graph with 30 vertices. The partial autocorrelation is shown for different time lags where the blue lines define a 95% confidence interval. If at lag k the partial autocorrelation is above or below such line, then we have confidence that with observed k samples in the past can predict the present. As shown in the figure, the Delta strategy is the one achieving the smallest autocorrelations. With $\varepsilon = 0$ we clearly get high autocorrelations at different lag points and as we proceed to $\varepsilon = 1$ we got the same profile of Delta. However, such lower predictability we obtain by increasing ε comes at a less informed strategy, which tends to distribute coverage equally to all the targets, without accounting for their importance.

6.2 Performance Against an Observer

The second experimental evaluation we present shows the performance of Delta and Idleness-ε against an attacker different from the worst-case one, that is an attacker that does not have access to the exact characterization of the inter-arrival times. We model it by using a simple local learning method for time series prediction as outlined in [7], namely a nearest neighbor predictor. Formally, the attacker at location l_j gets a sequence of observations defined as $O = (R_j^0, R_j^1, \ldots, R_j^t)$ where $O(i) = R_j^i$. The observer considers sub-sequences of finite length m (this parameter can be interpreted as the observer's finite memory). Thus, for $m - 1 \leq i < t$, call $O_i = (O(i - m + 1), \ldots, O(i))$. The observer then computes

$$i^* = \underset{i \in \{m,\ldots,t-1\}}{\arg\min} \; d(O_i, O_t)$$

and makes a prediction as $O(i^*+1)$. If $O(i^*+1) \geq a_j$ then an attack is attempted.

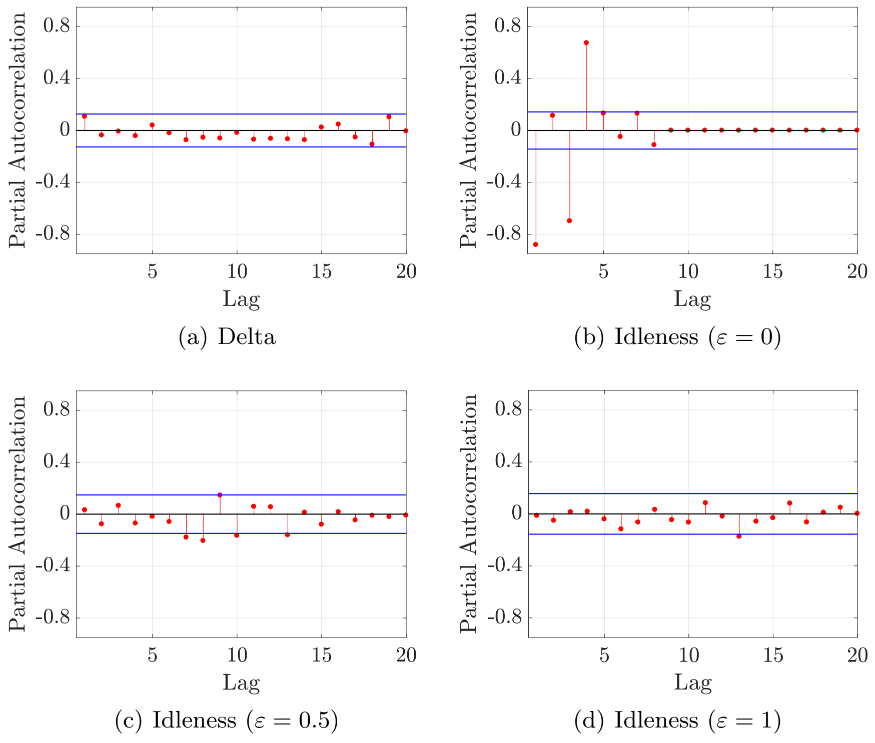

Fig. 3. Partial autocorrelation analysis for the time series of inter-arrival times at a target.

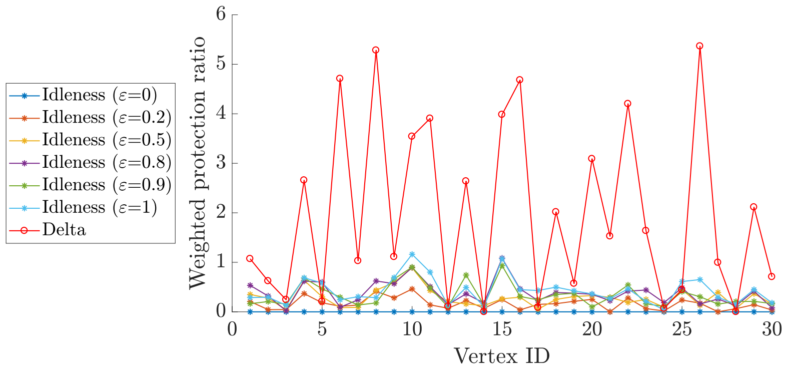

Fig. 4. Weighted protection ratio for different patrolling strategies.

In Figure 4, we depict the performance trend of the various strategies observed in a representative instance. For each vertex, we report the protection ratio defined as the number of unsuccessful attacks over the number of the attack attempts. The ratio is then multiplied by the value of the vertex it is referred to, to account for the fact that different vertices have different values. Each point in the curve can then be interpreted as the amount of protected value. As it can be seen, in this setting the ability of generating a sequence of inter-arrival times that is difficult to predict plays a critical role. However, increasing ε in the Idleness strategy introduces only marginal improvements while the Delta strategy achieves good performance in protecting the environment while, at the same time, misleading the observer. This performance gap is also summarized in Figure 5(a) showing the cumulative protected value (the sum over all the vertices of the weighted protection ratio) and Figure 5(b) where we report the ratio between the cumulative protected value of Delta and that achieved with different ε. To put these numbers into context, the protected value obtained by Delta is 56.19.

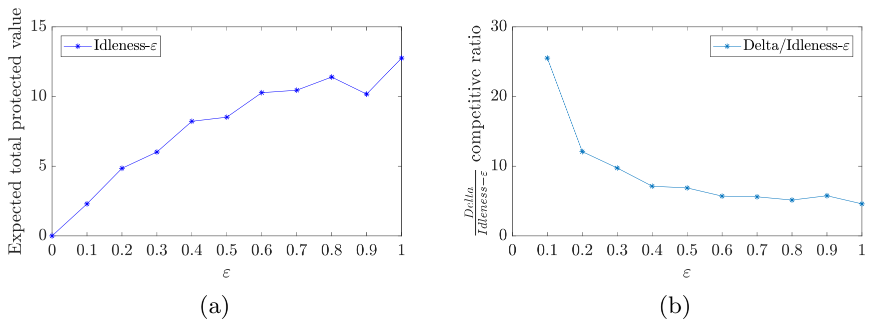

Fig. 5. (a) expected protected value for different ε values. (b) ratio between the protected value of the Delta and Idleness strategy.

7 Conclusions

In this paper we have studied a novel approach to implement patrolling strategies that are provably hard to learn for an observer. Our problem departs from classic literature in security games inasmuch as the attacker does not have complete knowledge of the patrolling strategy, but rather builds a model through observations limited to a fixed subset of the environment. We maintain that this approach is more relevant to practical applications than those based on worst case assumptions with an omniscient attacker. Our patrolling strategy essentially produces a white noise time series of interarrival times. Nevertheless, since the patroller does not know which vertex is being observed by the attacker, the strategy is biased towards visiting more often the vertices with high values and low attack times, i.e., those more likely to create large losses for the patroller. The ability of the patroller to introduce random bounded delays (as governed by the parameter Δ) when moving from vertex to vertex increases its unpredictability. It is important to stress that this patrolling approach is suited when the attacker learns through limited observations, but does not necessarily apply when the attacker has global visibility about the patroller's routes.

References

1. P. Agharkar, R. Patel, and F. Bullo. Robotic surveillance and Markov chains with minimal first passage time. In *Proc. of the IEEE conference on Decision and Control*, pages 6603–6608, 2014.
2. S. Alamdari, E. Fata, and S.L. Smith. Persistent monitoring in discrete environments: Minimizing the maximum weighted latency between observations. *The International Journal of Robotics Research*, 33(1):138–154, 2014.
3. S. Alpern, A. Morton, and K. Papadaki. Patrolling games. *Operations research*, 59(5):1246–1257, 2011.
4. B. An, M. Brown, Y. Vorobeychik, and M. Tambe. Security games with surveillance cost and optimal timing of attack execution. In *Int. Conf. on Autonomous Agents and Multi-agent Systems*, pages 223–230, 2013.
5. N. Basilico, N. Gatti, and F. Amigoni. Patrolling security games: Definition and algorithms for solving large instances with single patroller and single intruder. *Artificial Intelligence*, 184:78–123, 2012.
6. A. Blum, N. Haghtalab, and A.D. Procaccia. Lazy defenders are almost optimal against diligent attackers. In *AAAI*, pages 573–579, 2014.
7. G. Bontempi, S. Ben Taieb, and Y. Le Borgne. Machine learning strategies for time series forecasting. In *Business Intelligence*, pages 62–77. Springer, 2013.
8. Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302–308, 2004.
9. J. Grace and J. Baillieul. Stochastic strategies for autonomous robotic surveillance. In *Proc. of the IEEE Conference on Decision and Control*, pages 2200–2205, 2005.
10. R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.
11. A. Papoulis and S.U. Pillai. *Probability, Random Variables, and Stochastic Processes*. McGraw-Hill, 4th edition, 2002.
12. N. Privault. *Understanding Markov Chains*. Springer, 2013.
13. C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *Autonomous Robots*, 40(4):729–760, 2016.
14. S.M. Ross. *Introduction to Probability Models*. Elsevier, 2014.
15. R.S. Sutton and A.G. Barto. *Reinforcement Learning*. MIT Press, 1998.
16. M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.
17. William WS Wei. Time series analysis. In *The Oxford Handbook of Quantitative Methods in Psychology: Vol. 2*. 2006.
18. C. Zhang, V. Bucarey, A Mukhopadhyay, A. Sinha, Y. Qian, Y. Vorobeychik, and M. Tambe. Using abstractions to solve opportunistic crime security games at scale. In *Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 196–204, 2016.