Multirobot Patrolling Against Adaptive Opponents with Limited Information

Carlos Diaz Alvarenga Nicola Basilico Stefano Carpin

***Abstract*— We study a patrolling problem where multiple agents are tasked with protecting an environment where one or more adversaries are trying to compromise targets of varying value. The objective of the patrollers is to move between targets to quickly spot when an attack is taking place and then diffuse it. Differently from most related literature, we do not assume that attackers have full knowledge of the strategies followed by the patrollers, but rather build a model at run time through repeated observations of how often they visit certain targets. We study three different solutions to this problem. The first two partition the environment using either a fast heuristic or an exact method that is significantly more time consuming. The third method, instead does not partition the environment, but rather lets every patroller roam over the entire environment. After having identified strengths and weaknesses of each method, we contrast their performances against attackers using different algorithms to decide whether to attack or not.**

I. Introduction

Multirobot patrolling received considerable attention from the robotics community in the last decade. The literature encompasses models for different real-world domains and also provides algorithms to obtain patrolling strategies which guide a robot's visits to areas of an environment to be protected [14]. Many of these solutions are deployed as fully or partially autonomous multirobot surveillance systems [17].

One recently recognized dimension along which multirobot patrolling strategies can be compared concerns the use of standard optimization formulations versus the adoption of game theoretical frameworks that explicitly consider a rational and adversarial agent, often called *attacker*. The first kind of techniques typically seek policies that optimize a measure of effectiveness for the patrolling task. Commonly, idleness-based or frequency-based metrics are embraced over discrete settings represented with graphs. This line of works was started in [5] and is still currently active [16].

Techniques based on game-theoretical models usually come from the sub-field of 'security games' [19] where patrolling is performed considering a worst-case fully informed attacker. Building on top of these basic works, many recent contributions have relaxed the worst-case assumptions on the attacker in favor of more realistic settings. Examples can be found in the use of bounded rationality for the attacker [11] and, more relevantly, in the introduction of limited observation capabilities where it is assumed that the attacker can condition its decisions on a belief constructed by observing the realization of the patrolling strategy [1], [4]. In [2], we introduced a single-robot patrolling setting where the attacker can only gather information from a single location of the environment. This assumption adheres to many real-life situations where the costs of adversarial intelligence are prohibitive. In this work, we study the properties of the model proposed in [2] by extending it to the multirobot case.

When compared against their single-robot counterparts, multirobot settings introduce additional challenges. Scalability is a central one, since the computation of joint optimal policies is generally computationally expensive. Customary approaches try to shrink the strategy spaces by limiting the allowed coordination among robots at run time [3]. Conflicts among patrollers could arise [6] while environmental dynamics can pose the need for non-trivial adaptive task-reallocation methods [9]. Sometimes, online coordination among robots might not be even possible due, for example, to the lack of a suitable communication infrastructure [18].

Environment partitioning is one way to mitigate such problems by diverting part of the coordination issues to the offline dimensions. This basic *divide et impera* idea of allocating robots to sub-regions of the environment has been proven relevant in the multirobot patrolling domain (see, for example, [13], [10], [15]). We try to leverage such an idea to extend our novel patrolling model from [2] for multiple robots. We devise and evaluate two scenarios depending on whether partitions are used or not. We propose a heuristic and exact method to compute partitions and we carry out a theoretical analysis of how the patrolling performance is affected when more patrollers are concurrently deployed. Moreover, to address strengths and limitations of the partitioning methods we propose, we contrast these solutions with a solution where multiple robots independently patrol the whole graph without any sort of coordination or load balancing.

II. Patrolling setting definition

We embrace the usual graph-based setting adopting the model we recently studied in [2] for the single-robot case and introduce the presence of multiple patrollers.

We assume to have an environment where K target locations, *targets* for short, must be protected by means of repeated visits. Targets are denoted as $\{l_1, \ldots, l_K\}$. Their topological layout is described by a weighted undirected graph $G = (V, E, d)$ where $V = \{l_1, \ldots, l_K\}$ and $(l_i, l_j) \in E$ indicates that a patroller can move from l_i to l_j (or *vice versa*) in a time equal to d_{ij}. We assume that this graph is connected and we will always work on its transitive closure,

N. Basilico is with the Department of Computer Science , University of Milan, Milan, Italy. C. Diaz Alvarenga and S. Carpin are with the Department of Computer Science and Engineering, University of California, Merced, CA, USA.

that is if $(l_i, l_j) \notin E$ we add it setting d_{ij} to the length of the shortest path between l_i and l_j in the original graph. Each target l_i is characterized by a *value* v_i measuring its importance and an *attack time* a_i expressing the temporal cost needed to compromise it.

We consider $M \leq K$ *attackers*, each with the objective of compromising a given target. In order to successfully compromise a target l_i, an attacker must spend a time greater or equal to a_i on such a target without being detected by any patroller. Attacks are non-preemptive, meaning that they will terminate either with a success or a capture. The interaction between the team of patrollers and each single attacker can be thought as driven by a constant-sum revenue distribution where the patrollers get the total amount of protected value and the attacker gets the value of the target it compromises. Formally, patrollers and any attacker will get $(\sum_{i=1}^{K} v_i, 0)$ in case of a failed attack, and $(\sum_{i=1}^{K} v_i - v_j, v_j)$ in case of a successful attack on target l_j, respectively. We will not use such payoff structure in a game-theoretical model, but embrace the underlying interpretation where v_i can be seen as the value *stored* in l_i, while a_i encodes a measure of resiliency of the target.

To protect the targets, $N < K$ patrollers are deployed. We assume that when a patroller visits a target under attack it neutralizes the attacker: if one attacker attacks l_i at time t and the next visit to l_i by any of the patrollers occurs before time $t + a_i$, the attacker fails its attempt. If instead no patroller visits l_i before $t + a_i$, l_i is compromised.

The motion of the patrollers in the graph is governed by strategies that are not known to the attackers. However, through repeated observations, each attacker can construct a belief based on the following assumptions:

A) each attacker has local observation capabilities spatially confined to a single given target (as in [2]);
B) patrollers are indistinguishable;
C) attackers are unaware of each other.

Assumption A) allows each attacker to know the times at which a patroller arrives and leaves the target it is observing. This target is chosen before gathering any observation of the patrollers and hence it is not influenced by them. In other words, the strategic decision that the attacker has to perform is whether to attack a given target under observation or not. We assume that the M targets under observation are unknown to the patrollers. This assumption reflects our choice of locally limited observation capabilities: our attackers cannot gather observations from multiple targets and use such a knowledge to strategically choose which target to attack. Assumption B) means that an attacker observing a target can just see that one of the N patrollers is visiting but it cannot distinguish which one. Assumption C) implies that attackers cannot share the observations they gather, nor they can coordinate their decisions to maximize their chances.

As in [2], we assume that the patrolling strategy followed by the i-th patroller can be modeled by a time-invariant Markov chain described by a $K \times K$ transition matrix $\mathbf{P}^i$ whose entries are all strictly positive. As we will see in the following, we will consider both the case where all the patrollers follow the same strategy and the case where each patroller follows a different one. The (i, j) entry in $\mathbf{P}^i$ represents the probability that the i-th patroller will visit l_j after having visited l_i. When a patroller moves from l_i to l_j, it will take a time of at least d_{ij}. In [2] we introduced a new class of strategies characterized by *motion delays*. The main idea builds upon the assumption that when a patroller moves from i to j it will introduce a random traveling delay so that the overall transition time will be $d_{ij} + \zeta$ where ζ is a random variable drawn from a uniform distribution over $[0, \Delta]$. As we showed in [2], such random delays could make it more difficult for an attacker to precisely forecast the next visit to a target. As a result, these strategies outperform other non-delayed approaches against attackers that condition their attack on patroller observations they have collected on the observed target. The general structure of such delayed strategies for a patroller r is summarized in Algorithm 1.

```
1 Input: Transition Matrix P^i;
2 Select start vertex l_i ~ π^i ;
3 while true do
4     Select next vertex l_j with probability P^i_ij;
5     Generate random time t ~ U[d_{i,j}, d_{i,j} + Δ];
6     Move to l_j spending time t;
7     l_i ← l_j;
```

Algorithm 1: Time-delayed patrolling strategy (from [2])

The transition matrix $\mathbf{P}^i$ is assumed to be given and computed from an optimization taking the patrolling setting as input and returning the optimal stationary distribution $\boldsymbol{\pi}^i$ from which $\mathbf{P}^i$ can be reconstructed via the Metropolis-Hastings algorithm (see [2]). Notice that, since $\mathbf{P}^i$s have positive entries, each of them has a stationary distribution $\boldsymbol{\pi}^i$ (a K-dimensional vector such that $\sum_{j=1}^{K} \pi_j^i = 1$ and $\boldsymbol{\pi}^i = \boldsymbol{\pi}^i \mathbf{P}^i$.)

Against this background, we study adversarial patrolling when this is carried out by a team of N patrollers executing the above defined patrolling strategy over one or more subgraphs that span the whole environment. Patrollers execute their strategies independently, meaning that we do not consider any online coordination taking place. Instead, we allow *offline coordination*, seen as a subdivision of the efforts over different sub-regions of the environment, and how it can impact on the resulting level of protection. To do this, we will consider and compare two configurations:

- **Partitioned Patrollers**: the patrolling effort is distributed according to a partition of the environment (computed offline). Each patroller is assigned to one sub-region (sub-graph); the sub-regions do not overlap and their union covers the whole environment. Here offline coordination is performed.
- **Non-Partitioned Patrollers**: no patitioning is performed, each patroller potentially covers the whole environment. Here offline coordination is not performed. If patrollers do not move in sync, this approach will in

general shorten the time between successive visits to a vertex, thus making it harder for attackers to succeed.

Each configuration has its advantages and disadvantages. The partitioned case allows the patrollers to allocate more resources where needed. For example, the environment could be partitioned into small sub-regions when the total value of the covered targets is high. In general, the sub-regions allow frequent revisits, thus making it harder for an attacker to succeed. On the other hand, this approach is less robust to faults. If one of the patrollers were to stop working, the sub-region that was assigned to it would remain uncovered. The non-partitioned approach instead, is more resilient to failures because even if one patroller fails, all other patrollers can periodically revisit any location.

III. Partitioned patrollers

We start by considering the configuration where the environment is partitioned into N sub-graphs and each patroller is assigned to one of them. The rationale here is to balance as much as possible the workload among the patrollers.

A partition of $G = (V, E)$ can be represented with a set $\{V_1, V_2, \ldots, V_N\}$ where $V_i \subseteq V$, for any $i \neq j$ it holds that $V_i \cap V_j = \{\varnothing\}$, and such that $\cup_{i=1}^{n} V_i = V$. Given a partition element V_i, we denoted as $G[V_i]$ the subgraph induced by V_i on G. Such an induced subgraph has V_i as the set of vertices, the set of edges is given by $E_i = \{(l_i, l_j) \in E \text{ s.t. } l_i \in V_i \wedge l_j \in V_i\}$, and it represents the sub-region of the environment that will be assigned to the i-th patroller. To compute partitions, we need to quantify the patrolling workload. For any partition element V_i, we choose the cumulative temporal cost of the edges in E_i:

$$I(G[V_i]) = \sum_{(l_i,l_j)\in E_i} d_{ij}$$

Ideally, it is desirable to seek partition elements with low values of $I()$. A small cumulative temporal cost is an indicator that the total time needed to cover $G[V_i]$ can be limited. These features can ease the patrolling task on the environment's sub-region associated to $G[V_i]$: protecting an area where targets tend to be close to each other can ensure higher frequencies of visits and, hence, better protection. Clearly, this is an heuristic principle which does not guarantee any optimality on the patrolling performance and that does not constitute the only possible choice (for example, the maximum temporal cost could be an alternative.) We decided to opt for such a metric for its simplicity of definition and, more importantly, for the fact that it will allow us to devise two different methods for computing partitions.

Given $I()$, our partitioning problem can be stated as the following: compute a partition $\{V_1, V_2, \ldots, V_N\}$ of G such that $\max\{I(G_1), I(G_2), \ldots, I(G_N)\}$ is minimized. This problem shares core features with the well-known Graph Partition (decision) Problem (GPP) defined as follows: given a weighted, undirected graph $G = (V, E)$ is there a $V' \subset V$ such that $I(G[V']) = I(G[V \setminus V'])$? As shown in [7] (problem SP12), the GPP is $\mathcal{NP}$-complete from the *partition* problem. An immediate consequence of this result is that our partitioned patrollers problem is $\mathcal{NP}$-hard to solve optimally (the reduction is not shown for lack of space). In the following we introduce and evaluate two methods to solve this problem.

A. Multilevel Graph Partitioning

The first partitioning approach we propose is a heuristic based on the multi-level graph partitioning (MLGP) discussed in [8] and [12]. This method is divided into three steps: graph coarsening, partitioning, and uncoarsening. The final partition is both balanced in terms of the number of vertices in each sub-graph and also minimal with respect to the graph cut or inter-sub-graph cumulative edge weight. The balanced k-way minimum-cut partition performed by the MLGP algorithm does not directly solve our partitioning problem. Indeed, our formulation seeks partitions that minimize the cumulative weight of graph edges that, under the MLGP formulation, are left uncut. Because of this, we introduce two transformations of the edge weights.

The underlying idea in both methods is to re-write the edge weights in such a way that the cut minimization performed by MLGP will produce a partition similar to a division based on the cumulative weight of uncut edges (those that, in our problem, are summed up to obtain $I()$ in each partition element.) By inverting the edges weights of the graph, when minimizing the cut of the graph, the MLGP algorithm will remove edges with large weights, leaving those with small costs uncut (inside a partition element).

The first method for graph inversion is the following transformation of the graph edges:

$$d_{ij} = \left\lfloor \Big(\max_{(l_i,l_j)\in E} \{d_{ij}\} - d_{ij} + 1 \Big)^2 \right\rfloor \quad \forall l_i, l_j \in V$$

This transformation will leave the heaviest edges in the graph with the lowest weights, meaning that targets that in the original graph are very far apart will have, after the transformation, low edge weights. The k-way minimum-cut partition of the transformed graph will tend to keep vertices that were far apart out of the same sub-graph in the resulting partition. In other words, the edges in the minimum cut of the transformed graph will tend to have large weights in the original graph, which means that sub-graphs in the resulting partition will be characterized by low cumulative edge costs.

The second method for transforming the edge weights is based on a normalization of both the edge weights and each target's value-to-attack-time ratio defined, for a target l_i, as $\rho_i = v_i/a_i$. The normalization of these quantities is performed in this way:

$$d_{ij}^{norm} = \frac{d_{ij}}{\max\limits_{(l_i,l_j)\in E} \{d_{ij}\}}, \quad \rho_i^{norm} = \frac{\rho_i}{\max\limits_{l_i \in V} \{\rho_i\}} \quad \forall l_i, l_j \in V$$

Then we consider, for each edge, its normalized weight multiplied by the average normalized value-to-attack-time ratios

of the two associated vertices. Thanks to the normalization, this operation will return a value between 0 and 1:

$$d_{ij} = d_{ij}^{norm} \frac{p_i^{norm} + \rho_j^{norm}}{2}$$

Finally the values are scaled by an arbitrary factor s and the ceiling is taken to avoid null weights:

$$d_{ij} = \lceil (s - s \cdot d_{ij} + 1)^2 \rceil$$

The idea behind this second transformation is to integrate in the new computed weights also a bias related to the value-to-attack-time ratios of the targets connected by an edge. The value ρ_i provides a measure of the *critical level* of a target. The higher the target's value the more critical it is because if it gets compromised the value loss will be large. Similarly, the lower the attack time the easier will be for the attacker to compromise that target. As a consequence, this second transformation not only will induce MLGP to push targets that are far apart into different sub-graphs, but also to try to separate targets that have a high critical level.

B. MILP-based partitioning

The second approach we introduce is an exact method based on the resolution of a Mixed Integer Linear Program (MILP). The following decision variables denote, for any sub-graph $G[V_i]$ induced by a partition element V_i, whether any target and edge of the original graph belong to that sub-graph, respectively. Formally, these variables are:

$$x_i^h = \begin{cases} 1 & \text{if } l_i \in V_h \\ 0 & \text{otherwise} \end{cases}, \quad y_{i,j}^h = \begin{cases} 1 & \text{if } (l_i, l_j) \in E_h \\ 0 & \text{otherwise} \end{cases}$$

Then the binary linear program reads as follows:

$$\min u \quad \text{s.t.} \tag{1}$$

$$\sum_{h \in \{1,\dots,N\}} x_i^h = 1 \qquad \forall l_i \in V \tag{2}$$

$$\sum_{i \in V} x_i^h \geq \gamma \qquad \forall h \in \{1, \dots, N\} \tag{3}$$

$$y_{i,j}^h \leq x_i^h \qquad \forall l_i, l_j \in V \tag{4}$$

$$y_{i,j}^h \leq x_j^h \qquad \forall l_i, l_j \in V \tag{5}$$

$$y_{i,j}^h \geq x_i^h + x_j^h - 1 \qquad \forall l_i, l_j \in V, h \in \{1, \dots, N\} \tag{6}$$

$$u \geq \sum_{l_i, l_j \in V} d_{i,j} y_{i,j}^h \qquad \forall h \in \{1, \dots, N\}, i > j \tag{7}$$

$$x_i^h \in \{0, 1\} \qquad \forall l_i \in V, h \in \{1, \dots, N\} \tag{8}$$

Constraints (2) force each target to belong to exactly one sub-graph in the partition. Constraints (3) require each sub-graph to include at least γ targets (in general $\gamma \geq 2$, and in our experiments we set $\gamma = 2$). This requirement translates to a minimum workload assigned to each patroller, trying to avoid situations in which one patrollers stays fixed on a give target as such a situation would be equivalent to removing that target and one patroller from the original problem. Constraints (4), (5), and (6) bound the x and y decision variables. These three set of constraints express, in a linearized form, the fact that and edge (l_i, l_j) belongs to a sub-graph if and only if both l_i and l_j belong to that graph too. Constraints (7) define the auxiliary decision variable u and any upper bound over all the cumulative temporal costs of the various sub-graphs in the partition. The objective function minimization provided in (1) requires to seek the minimum upper bound. Constraints (8) impose a binary integrality to the xs (these are the only integrality constraints needed since the joint effect of Constraints (4), (5), and (6) and the minimization (1) guarantee that, at the optimum, the ys always get a binary value).

IV. Non-partitioned patrollers

In this second configuration, we consider no offline coordination and also N patrolling units independently executing the same Markov strategy $\mathbf{P}$ (computed, like before, as described in [2]) over the whole graph. In this case no partition of the environment is computed.

One key aspect we aim at studying here is how the expected return times to a generic target l_j vary when the $N \geq 2$ patrollers are deployed. Indeed, such return times are at the core of the performances obtained by $\mathbf{P}$ against an attacker that tries to learn them by observing a single target and, when a confident estimate is reached, uses such knowledge to determine whether to attack or not. The return time to a target l_j when N patrollers use the same strategy can be formally defined as follows. For $1 \leq i \leq N$, let x_t^i be the state occupied by the i-th patroller at time t (this can correspond to being at some target or traveling along some edge). The return time to a target l_j of any of the N patrollers is then defined as the following random variable:

$$r_j^N = \inf\{k \text{ s.t. } \exists i, w, t \ x_{t+k}^i = x_t^w = l_j\}$$

In particular, we would like to determine how $\mathbb{E}[r_j^N]$ relates to $\mathbb{E}[r_j]$ (the expected return time with a single patroller).

Consider a Markov chain with K states and collapse it into a two-state chain where one state corresponds to state j in the original chain and state ξ represents the aggregation of all the other states different from j.

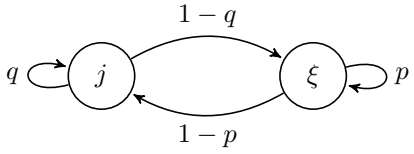

Fig. 1. The collapsed two-states Markov chain.

Figure 1 depicts such a collapsed Markov Chain whose transition matrix is:

$$\mathbf{P_{coll}} = \begin{bmatrix} q & 1-q \\ 1-p & p \end{bmatrix}$$

Here, q denotes the transition probability from state j to itself and p will denote the transition probability from any state

different from j (denoted by ξ) and remaining there. The expected return time to state j is given by:

$$\mathbb{E}[r_j] = \frac{(1-q)+(1-p)}{(1-p)}$$

Next, we can derive the relation between the values p, q and the non-collapsed transition matrix $\mathbf{P}$. The value q is $\mathbf{P}_{jj}$ as it corresponds to the probability of remaining in state j. To obtain p we solve for the value $1-p$. Using the total probability law $1-p$ can be derived as:

$$1-p = \sum_{i \neq j} \mathbf{P}_{ij}\mathbf{P}_i^{'}, \text{ where } \mathbf{P}_i^{'} = \frac{\pi_i}{\sum_{i \neq j} \pi_i}$$

which corresponds to the probability of being in state ξ given that we are not in state j. Thus, the expected return time for any state in the new two-state Markov chain can be explicitly computed from the values in the original $K \times K$ transition matrix. Given this, we can now provide a characterization of the expected return times at the targets in the multi-patroller setting using the transition matrix $\mathbf{P}$ adopted by each of the N patrollers on the graph. We again formulate the multi-patroller problem as a two-state Markov chain. We will call *covered* the state where at least one patroller is at the target l_j, while *uncovered* is the state when no patrollers are currently visiting l_j. The transition matrix associated to this Markov chain is:

$$\mathbf{P}_{\mathbf{coll}} = \begin{bmatrix} q^* & 1-q^* \\ 1-p^* & p^* \end{bmatrix}$$

and

$$\mathbb{E}[r_j^N] = \frac{(1-q^*)+(1-p^*)}{(1-p^*)}$$

The value q^* in the transition matrix is associated with the probability that given there is at least one patroller on target l_j, at least one of them remains in that target in the consecutive transition. Next, p^* corresponds to the probability that given that none of the patrollers are at target l_j, all of them remain outside of l_j in the following transition. We can reason that p^* is equal to p^N, because p is the probability that one patroller in state ξ remains in the same state after one transition. Moreover, it can be shown that (proof omitted for brevity):

$$(1-q^*) = \sum_{k=1}^{N} \binom{N}{k} \eta \pi_j^k (1-\pi_j)^{N-k} (1-q)^k p^{N-k}$$

$$\eta = \frac{1}{1-(1-\pi_j)^N}.$$

V. Evaluation

We provide an empirical evaluation of the patrolling strategies obtained with our proposed techniques: heuristic partitions without critical levels (non-weighted, PNW) and with critical levels (weighted, PW), partitions computed with our exact method (MIP, run with a deadline of 30 minutes), and non-partitioned patrollers (NP). Varying the number of vertices in the graph from 30 to 60 (with increments of 10), for each size we generate 10 random instances and we run patrolling missions over the obtained graphs where each patroller visits 10,000 vertices. Due to limited space, we present only the case where the number of patrollers is either 5 or 10, even though we have explored a larger range.

We assumed one attacker observing each target and considered two different types of attackers. The first type, called Maximum-Likelihood (ML) attacker, fits an exponential distribution over the inter-arrival times observed at its target. Then, it generates a prediction for the next inter-arrival time by computing the expectation of that distribution. The second type of attacker, called Nearest-Neighbor attacker (NN), treats the sequence of inter-arrival times as a temporal series and uses a nearest neighbor over the last 10 observed elements from the series to forecast the next one. In both cases, at each iteration where the predicted inter-arrival time is larger than the target's attack time, the attacker decides to attack. As it is evident, the patrollers could perform more frequent visits but should also try to induce an overestimation of the next attack time.

The plots will show the *protection ratio* defined as the number of the attacks intercepted by a patroller divided by the attempted ones and multiplied by the value of the target. We show also the number of attack attempts induced by a patrolling strategy and, for each l_i, the value $v_i \cdot Pr[r_u > a_i]$ which provides an indicator of the protected value not dependent on how the attacker estimates arrival times (called *intrinsic loss* in the following). Each chart features on the y axis the average quantity over all the vertices of the graph while on the x axis we have the single instances from the smallest to the biggest K (number of targets).

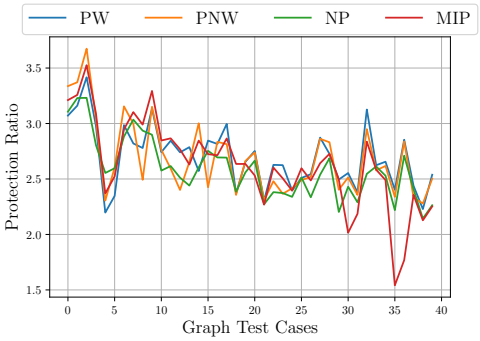

Fig. 2. Protection ratio for five patrollers against a ML attacker.

Figures 2 and 3 show the average protection ratios with 5 patrollers against the two types of attackers. The generally decreasing trends reflect the increasing difficulty of protecting a graph where there are more targets but the number of patrollers stays the same. The NP approach seems to be slightly dominated by the others suggesting that partitioning could be profitable. The two heuristics kept pace with the MIP method, even outperforming it at times when this last met the deadline without finding the optimal partition and returning, instead, the current best solution found.

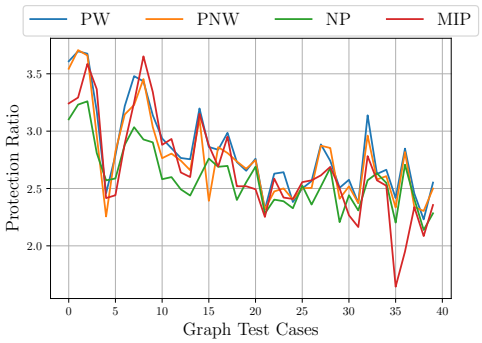

Fig. 3. Protection ratio for five patrollers against a NN attacker.

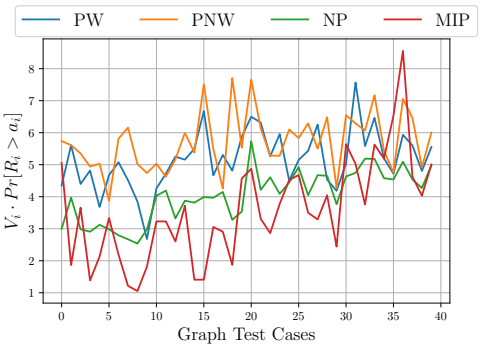

Fig. 4. Intrinsic loss for the case of 10 patrollers.

A more insightful analysis of these trends can be extracted if one considers not only the protection ratio, but also the number of attempted attacks – something we analyze for the case of 10 patrollers discussed next. With 10 patrollers things become more challenging and slightly more evident gaps start to emerge as shown in Figure 4 where we show the intrinsic loss for 10 patrollers. The figure clearly shows that the MIP method provides better coverage to vertices with high value. Figures 5 and 6 show the protection ratio against the two type of attackers. To fully appreciate the meaning of Figure 5, where it seems that the MIP method is underperforming, it is useful to consider Figure 7 where we show the number of attempted attacks. The figure shows that the MIP strategy induces the attacker to reduce its number of attempts. In general, partitioned-based methods induce fewer attack attempts, especially in those instances where computing a balanced partition is easier. This is reasonable since a well-balanced partition (recall the definition of $I()$) favours an even distribution of values and small inter-arrival times. This acts as a deterrent on the attacker that in many occasions opts for not attempting an attack. Thus, if based on the protection ratio partitioned strategies seem almost comparable to the non-partitioned one in terms of capturing the attacker, they clearly work better in keeping the attacker out. Taking all into considerations, the MIP strategy is the best one. While it is more demanding, our experiments show that using the approximate solution produced upon stopping the method after 30 minutes still provides good results. PW and PNW can be considered as fairly effective heuristics since their performance was not remarkably worse than MIP's. Their lower computational burden can be leveraged when scaling to very large patrolling settings.

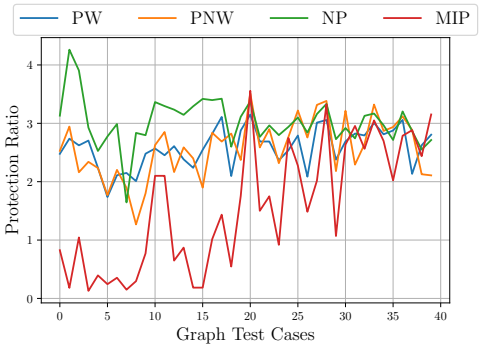

Fig. 5. Protection ratio for ten patrollers against a ML attacker.

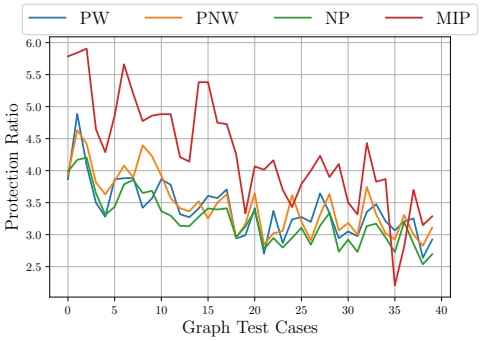

Fig. 6. Protection ratio for ten patrollers against a NN attacker.

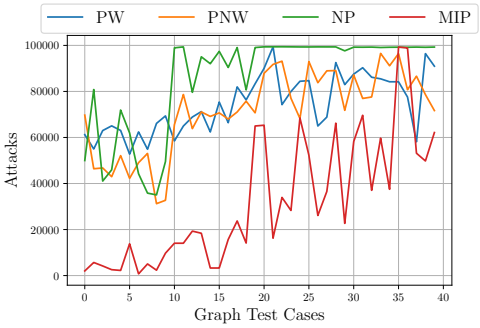

Fig. 7. Number of attacks for 10 patrollers against a ML attacker.

VI. Conclusions

In this paper, we studied three approaches for multirobot patrolling against an attacker that, through repeated observations, tries to predict when it is the best time to attack. Our method based on a MIP formulation turns out to be the best one, even when it is stopped before the optimal solution is found. For the two types of attacker we considered, this strategy discourages the attacker from attempting to compromise the assets being protected. In future works, we shall expand the analysis presented in this paper to consider refined models of the attacker behavior including,

for example, coordination between multiple attackers.

REFERENCES

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

[2] N. Basilico and S. Carpin. Balancing unpredictability and coverage in adversarial patrolling settings. In *Proceedings of the 2018 Workshop on Algorithmic Foundations or Robotics*, 2019 (to appear).

[3] N. Basilico, A. Celli, G. De Nittis, and N. Gatti. Coordinating multiple defensive resources in patrolling games with alarm systems. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, pages 678–686, 2017.

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

[5] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. In *Proc. IAT*, pages 302–308, 2004.

[6] L. Freda, Mario Gianni, F. Pirri, A. Gawel, R. Dubé, R. Siegwart, and C. Cadena. 3D multi-robot patrolling with a two-level coordination strategy. *Autonomous Robots*, 43(7):1747–1779, 2019.

[7] M.R Garey and D.S. Johnson. *Computers and Intractability. A guide to the theory of NP-Completeness*. W.H. Freeman and Company, 1979.

[8] G. Karypis and V. Kumar. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM Journal on Scientific Computing*, 20(1):359–392, 1998.

[9] N. Li, M. Li, Y. Wang, D. Huang, and W. Yi. Fault-tolerant and self-adaptive market-based coordination using hoplites framework for multi-robot patrolling tasks. In *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, pages 514–519, 2018.

[10] C. Pippin, H. Christensen, and L. Weiss. Performance based task assignment in multi-robot patrolling. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, pages 70–76, 2013.

[11] J. Pita, M. Jain, M. Tambe, F. Ordóñez, and S. Kraus. Robust solutions to stackelberg games: Addressing bounded rationality and limited observations in human cognition. *ARTIF INTELL*, 174(15):1142–1171, 2010.

[12] D. Portugal and R. Rocha. MSP Algorithm : Multi-Robot Patrolling based on Territory Allocation using Balanced Graph Partitioning. *ACM Symposium on Applied Computing*, pages 1271–1276, 2010.

[13] D. Portugal and R. Rocha. Msp algorithm: Multi-robot patrolling based on territory allocation using balanced graph partitioning. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1271–1276, 2010.

[14] D. Portugal and R. Rocha. A survey on multi-robot patrolling algorithms. In *Proc. DoCEIS*, pages 139–146, 2011.

[15] D. Portugal and R. P. Rocha. Performance estimation and dimensioning of team size for multirobot patrol. *IEEE Intelligent Systems*, 32(6):30–38, 2017.

[16] Y. Rizk, M. Awad, and E. W. Tunstel. Cooperative heterogeneous multi-robot systems: A survey. *ACM Comput. Surv.*, 52(2):29:1–29:31, 2019.

[17] C. Robin and S. Lacroix. Multi-robot target detection and tracking: taxonomy and survey. *AUTON ROBOT*, 40(4):729–760, 2016.

[18] M. Romeo, J. Banfi, N. Basilico, and F. Amigoni. Multirobot persistent patrolling in communication-restricted environments. In *Distributed Autonomous Robotic Systems: The 13th International Symposium*, pages 59–71, 2018.

[19] M. Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge University Press, 2011.