Minimizing Expected Intrusion Detection Time in Adversarial Patrolling

David Klaška  
Masaryk University  
Brno, Czechia  
david.klaska@mail.muni.cz

Antonín Kučera  
Masaryk University  
Brno, Czechia  
tony@fi.muni.cz

Vít Musil  
Masaryk University  
Brno, Czechia  
musil@fi.muni.cz

Vojtěch Řehák  
Masaryk University  
Brno, Czechia  
rehak@fi.muni.cz

ABSTRACT

In adversarial patrolling games, a mobile Defender strives to discover intrusions at vulnerable targets initiated by an Attacker. The Attacker’s utility is traditionally defined as the probability of completing an attack, possibly weighted by target costs. However, in many real-world scenarios, the actual damage caused by the Attacker depends on the time elapsed since the attack’s initiation to its detection. We introduce a formal model for such scenarios, and we show that the Defender always has an optimal strategy achieving maximal protection. We also prove that finite-memory Defender’s strategies are sufficient for achieving protection arbitrarily close to the optimum. Then, we design an efficient strategy synthesis algorithm based on differentiable programming and gradient descent. We evaluate the efficiency of our method experimentally.

KEYWORDS

Strategy synthesis, Security Games, Adversarial Patrolling

1 INTRODUCTION

This paper follows the security games line of work studying optimal allocation of limited security resources for achieving optimal target coverage [32]. Practical applications of security games include the deployment of police checkpoints at the Los Angeles International Airport [29], the scheduling of federal air marshals over the U.S. domestic airline flights [33], the arrangement of city guards in Los Angeles Metro [15], the positioning of U.S. Coast Guard patrols to secure selected locations [4], and also applications to wildlife protection [16, 34, 35].

Patrolling games are a special type of security games where a mobile Defender moves among protected targets with the aim of detecting possible incidents. Compared with static monitoring facilities such as sensor networks or surveillance systems, patrolling is more flexible and less costly on implementation and maintenance. Due to these advantages [36], patrolling is indispensable in detecting crimes [13, 19], managing disasters [26], wildlife protection [34, 35], etc. Many works consider human Defenders such as police squads or rangers [34] where the patrolling horizon is bounded. Recent technological advances motivate the study of robotic patrolling with unbounded horizon where the Defender is an autonomous device operating for a long time without interruption.

Most of the existing patrolling models can be classified as either regular or adversarial. Regular patrolling can be seen as a form of surveillance where the Defender aims at discovering accidents as quickly as possible by minimizing the time lag between two consecutive visits for each target. Here, a Defender’s strategy is typically a single path or a cycle visiting all targets. In adversarial patrolling, the Defender strives to protect the targets against an Attacker exploiting the best attack opportunities maximizing the damage. The solution concept is typically based on Stackelberg equilibrium [31, 37], where the Defender commits to a strategy \( y \) and the Attacker follows by selecting a strategy \( \pi \) maximizing the expected Attacker’s utility against \( y \). Defender’s strategies are typically randomized so that the Attacker cannot foresee the next Defender’s moves, and the Defender aims at maximizing the probability of discovering an attack before its completion. The adversarial model is also appropriate in situations when a certain protection level must be guaranteed even if the accidents happen at the least convenient moment.

In infinite-horizon adversarial patrolling models, every target \( \tau \) is assigned a finite resilience \( d(\tau) \), and an attack at \( \tau \) is discovered if the Defender visits \( \tau \) in the next \( d(\tau) \) time units. Although this model is adequate in many scenarios, it is not applicable when the actual damage depends on the time elapsed since initiating the attack. For example, if the attack involves setting a fire, punching a hole in a fuel tank, or setting a trap, then the associated damage increases with time. In this case, the Defender should aim at minimizing the expected attack discovery time rather than maximizing the probability of visiting a target before a deadline. We refer to Section 2.3 for a more detailed discussion.

In this work, we formalize the objective of minimizing the expected attack discovery time in infinite-horizon adversarial patrolling, and we design an efficient strategy synthesis algorithm. We start by fixing a suitable formal model. The terrain is modeled by the standard patrolling graph, and the Defender’s/Attacker’s strategies are also defined in the standard way. However, the expected damage caused by attacking a target \( \tau \) is defined as the expected time of visiting \( \tau \) by the Defender since initiating the attack, multiplied by the target cost \( \alpha(\tau) \). Intuitively, \( \alpha(\tau) \) is the “damage per time unit” when attacking \( \tau \). We use Stackelberg equilibrium as the underlying solution concept, and define the protection value of a
given Defender’s strategy \( y \) as the expected attack discovery time (weighted by target costs) guaranteed by \( y \) against an arbitrary Attacker’s strategy.

In general, a Defender’s strategy \( y \) may randomize and the choice of the next move may depend on the whole history of moves. The randomization is crucial for increasing protection value (a concrete example is given in Section 2.3). Since general strategies are not finitely representable, they are not algorithmically workable. Recent results on infinite-horizon adversarial patrolling [22, 23] identify the subclass of regular Defender’s strategies as sufficiently powerful to maximize the probability of timely attack discovery. Here, a strategy is regular if it uses finite memory and rational probability distributions. However, it is not clear whether regular strategies are equivalently powerful as general strategies when minimizing the expected attack discovery time. Perhaps surprisingly, we show that the answer is positive, despite all issues caused by specific properties of this objective. More precisely, we prove that the limit protection value achievable by regular strategies is the same as the limit protection value achievable by general strategies. This non-trivial result is based on deep insights into the structure of (sub)optimal Defender’s strategies.

Our second main contribution is an algorithm synthesizing a regular Defender’s strategy and its protection value for a given patrolling graph. We show that the protection value of a regular strategy is a differentiable function, and we proceed by designing an efficient strategy improvement procedure based on gradient descent.

We evaluate our algorithm experimentally on instances of considerable size. Since our work initiates the study of infinite-horizon adversarial patrolling with the expected attack discovery time, there is no baseline set by previous works. To estimate the quality of the constructed regular strategies, we consider instances where the optimal protection value can be determined by hand, but constructing the associated Defender’s strategy is sufficiently tricky to properly examine the capabilities of our strategy synthesis algorithm.

The experiments also show that our algorithm is sufficiently fast for recomputing a patrolling strategy dynamically when the underlying patrolling graph changes due to unpredictable external events. Hence, the applicability of our results is not limited just to static scenarios.

Our main contribution can be summarized as follows:

- We propose a formal model for infinite-horizon adversarial patrolling where the damage caused by attacking a target depends on the time needed to discover the attack.
- We prove that regular strategies can achieve the same limit protection value as general strategies.
- We design an efficient algorithm synthesizing a regular Defender’s strategy for a given patrolling graph, and we evaluate its functionality experimentally.

1.1 Related work

The literature on regular and adversarial patrolling is rich; the existing overviews include [3, 18, 30]. We give a summary of previous results on infinite-horizon adversarial patrolling, which is perhaps closest to our work.

Most of the existing results concentrate on computing an optimal moving strategy for certain topologies of admissible moves. The underlying solution concept is the Stackelberg equilibrium [31, 37], where the Defender/Attacker play the roles of the Leader/Follower.

For general topologies, the existence of a perfect Defender’s strategy discovering all attacks in time is \( \text{PSPACE} \)-complete [17]. Consequently, computing an optimal Defender’s strategy is \( \text{PSPACE} \)-hard. Moreover, computing an \( \epsilon \)-optimal strategy for \( \epsilon \leq 1/2n \), where \( n \) is the number of vertices, is NP-hard [24]. Hence, no feasible strategy synthesis algorithm can guarantee (sub)optimality for all inputs, and finding high-quality strategy in reasonable time is challenging.

The existing strategy synthesis algorithms are based either on mathematical programming, reinforcement learning, or gradient descent. The first approach suffers from scalability issues caused by non-linear constraints [5, 6]. Reinforcement learning has so far been successful mainly for patrolling with finite horizon, such as green security games [10, 20, 34, 35]. Gradient descent techniques for finite-memory strategies [22, 23, 25] are applicable to patrolling graphs of reasonable size.

Patrolling for restricted topologies has been studied for lines, circles [1, 2], or fully connected environments [12]. Apart from special topologies, specific variants and aspects of the patrolling problem have been studied, including moving targets [11, 14], multiple patrolling units [8], movement of the Attacker on the graph [7], or reaction to alarms [9, 27].

2 THE MODEL

In this section we introduce a formal model for infinite-horizon patrolling where the Defender aims at minimizing the expected attack discovery time. The terrain (protected area) is modeled by the standard patrolling graph [5, 6, 22, 23]. We use the variant where time is spent by performing edges [22, 23] rather than by staying in vertices [5, 6]. The model of Defender/Attacker is also standard [24, 25] (the study of patrolling models usually starts by considering the scenario with one Defender and one Attacker, and we stick to this approach). The new ingredient of our model in the way of evaluating the protection achieved by Defender’s strategies (see Section 2.3), and here we devote more space to explaining and justifying our definitions.

In the rest of this paper, we use \( \mathbb{N} \) and \( \mathbb{N}_+ \) to denote the sets of non-negative and positive integers. We assume familiarity with basic notions of probability, Markov chain theory, and calculus.

2.1 Terrain model

Locations where the Defender selects the next move are modeled as vertices in a directed graph. The edges correspond to admissible moves, and are labeled by the corresponding traversal time. The protected targets are a subset of vertices with integer weights representing their costs. Formally, a patrolling graph is a tuple \( G = ( V, T, E, tm, \alpha) \) where

- \( V \) is a finite set of vertices (Defender’s positions);
- \( T \subseteq V \) is a non-empty set of targets;
- \( E \subseteq V \times V \) is a set of edges (admissible moves);
- \( tm : E \rightarrow \mathbb{N}_+ \) specifies the traversal time of an edge;
- \( \alpha : T \rightarrow \mathbb{R}_+ \) defines the costs of targets.
We require that $G$ is strongly connected, i.e., for all $v, u \in V$ there is a path from $v$ to $u$. We write $u \rightarrow v$ instead of $(u, v) \in E$, and use $\sigma_{\text{max}}$ to denote the maximal target cost.

The set of all non-empty finite and infinite paths in $G$ are denoted by $\mathcal{H}$ (histories) and $\mathcal{W}$ (walks), respectively. For a given history $h = v_1, \ldots, v_n$, we use $tm(h) = \sum_{i=1}^{n-1} tm(v_i, v_{i+1})$ to denote the total traversal time of $h$.

## 2.2 Defender and Attacker

We adopt a simplified patrolling scenario with one Defender and one Attacker. In the rest of this section, let $G$ be a fixed patrolling graph.

### 2.2.1 Defender

A Defender’s strategy is a function $\gamma$ assigning to every history $h \in \mathcal{H}$ of Defender’s moves a probability distribution on $V$ such that $\gamma(h)(v) > 0$ only if $hv \in \mathcal{H}$, i.e., $u \rightarrow v$ where $u$ is the last vertex of $h$. We also use walk$(h)$ to denote the set of all walks initiated by a given $h \in \mathcal{H}$.

For every initial vertex $v$ where the Defender starts patrolling, the strategy $\gamma$ determines the probability space $(\mathcal{W}, \mathcal{F}, \mathbb{P}^\gamma)$ over the walks in the standard way, i.e., $\mathcal{F}$ is the $\sigma$-field generated by all walk$(h)$ where $h \in \mathcal{H}$, and $\mathbb{P}^\gamma$ is the unique probability measure satisfying $\mathbb{P}^\gamma(\text{walk}(h)) = \prod_{i=1}^{n-1} \gamma(v_i, v_i)(v_{i+1})$ for every history $h = v_1, \ldots, v_n$ where $v_1 = v$ (if $v_1 \neq v$, we have that $\mathbb{P}^\gamma(\text{walk}(h)) = 0$). We use $\mathbb{P}^\gamma[R]$ to denote the expected value of a random variable $R$ in this probability space.

### 2.2.2 Attacker

The Attacker observes the history of Defender’s moves and decides whether and where to initiate an attack. In general, the Attacker may be able to determine the next Defender’s move right after the Defender leaves the vertex currently visited. For the Attacker, this is the best moment to attack, because initiating an attack in the middle of a Defender’s move can only decrease the Attacker’s utility (cf. Section 2.3).

Formally, an observation is a sequence $o = v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$, where $v_1, \ldots, v_n$ is a path in $G$. Intuitively, $v_1, \ldots, v_n$ is the sequence of vertices visited by the Defender, $v_n$ is the currently visited vertex, and $v_n \rightarrow v_{n+1}$ is the edge taken next. The set of all observations is denoted by $\Omega$.

An Attacker’s strategy is a function $\pi: \Omega \rightarrow \{\text{wait}, \text{attack}_1 : \tau \in T\}$. We require that if $\pi(v_1, \ldots, v_n, v_n \rightarrow u_1, u_n \rightarrow v_{n+1}) = \text{attack}_\tau$ for some $\tau \in T$, then $\pi(v_1, \ldots, v_i, u_i \rightarrow u_{i+1}) = \text{wait}$ for all $1 \leq i < n$, i.e., the Attacker exploits an optimal attack opportunity.

## 2.3 Protection value

Suppose the Defender commits to a strategy $\gamma$ and the Attacker selects a strategy $\pi$. The expected damage caused by $\pi$ against $\gamma$ is the expected time to discover an attack scheduled by $\pi$ weighted by target costs.

More precisely, we say that a target $\tau$ is attacked along a walk $w = v_1, v_2, \ldots$ if $\pi(v_1, v_2, v_n, v_n \rightarrow v_{n+1}) = \tau$ for some $n$. Note that the index $n$ is unique if it exists. Let $m > n$ be the least index such that $v_m = \tau$. If no such $m$ exists, we say that the attack along $w$ is not discovered. Otherwise, the attack is discovered in time $tm(v_n, \ldots, v_m)$.

Let $\mathcal{D}^\pi: \mathcal{W} \rightarrow \mathbb{N}_\infty$ be a function defined as follows:

$$
\mathcal{D}^\pi(w) = \begin{cases}
\ell \cdot \alpha(\tau) & \text{if } \tau \text{ is attacked along } w \text{ and the attack is discovered in time } \ell; \\
\infty & \text{if } \tau \text{ is attacked along } w \text{ and the attack is not discovered; } \\
0 & \text{if no target is attacked along } w.
\end{cases}
$$

The expected damage caused by $\pi$ against $\gamma$ initiated in $v$ is defined as $\mathbb{E}^\gamma[\mathcal{D}^\pi]$. Since the Defender may choose the initial vertex $v$, we define the protection value achieved by $\gamma$ and the limit protection value as follows:

$$
\text{Val}(\gamma) = \min_v \sup_{\pi} \mathbb{E}^\gamma[\mathcal{D}^\pi]
$$

(1)

$$
\text{Val} = \inf_{\gamma} \text{Val}(\gamma)
$$

(2)

We say that a Defender’s strategy $\gamma$ is optimal if $\text{Val}(\gamma) = \text{Val}$.

### 2.3.1 Discussion

In this section, we discuss possible alternative approaches to formalizing the objective of discovering an initiated attack as quickly as possible.

Note that this objective is implicitly taken into account in regular patrolling where the Defender aims at minimizing the time lag between two consecutive visits for each target (see Section 1). For randomized strategies, one may try to minimize the expected time lag between two consecutive visits for each target. At first glance, this objective seems similar to minimizing $\sup_{\gamma} \mathbb{E}^\gamma[\mathcal{D}^\pi]$. In reality, the objective is different and problematic. To see this, consider the trivial patrolling graph of Fig. 1a with two targets $v_1, v_2$ and four edges (incl. two self-loops) with traversal time 1. The costs of both targets are equal to 1. A natural strategy $\gamma_1$ for patrolling these targets is a deterministic loop alternately visiting $v_1$ and $v_2$ (see Fig 1b). Then, the maximal expected time lag between two consecutive visits of a target is 2, and we also have that $\text{Val}(\gamma_1) = 2$. However, consider the randomized strategy $\gamma_2$ of Fig. 1c. In the target currently visited, the Defender performs the self-loop with probability 0.99, and with the remaining probability 0.01, the Defender moves to the other target (see the dashed arrows in Fig. 1c). For $\gamma_2$, the maximal expected time lag between two consecutive visits of a target is again equal to 2 (in Markov chain terminology, the stationary distribution determined by $\gamma_2$ assigns 1/2 to each target, and hence the mean recurrence time is equal to 2 for each target). Hence, if we adopted minimizing the maximal expected time lag between two consecutive visits of a target as the Defender’s objective, the strategies $\gamma_1$ and $\gamma_2$ would be equivalently good, despite the fact that the expected time to visit $v_2$ from $v_1$ is 100 when the Defender commits to $\gamma_2$. Observe that the difference between $\gamma_1$ and $\gamma_2$ is captured properly by our approach.

Let us note that although the example of Fig. 1 contains self-loops (which do not appear in real-world patrolling graphs), these can easily be avoided by inserting auxiliary vertices so that the demonstrated deficiency is still present.

One may still argue that the above problem is caused just by allowing the Defender to use randomized strategies. This is a valid objection. So, the question is whether randomization is really needed, i.e., whether the Defender can achieve strictly better protection by using randomized strategies.
Consider the graph of Fig. 2 with two targets \( \tau_1 \) and \( \tau_2 \) where \( \alpha(\tau_1) = 1, \alpha(\tau_2) = 2 \), and the traversal time of every edge is 1.

The protection achieved by an arbitrary deterministic strategy is not better than 8, because the Attacker can wait until the Defender starts moving from \( \tau_2 \) to \( \tau_1 \) so that the next move selected after arriving to \( \tau_1 \) will be edge leading to \( \tau_1 \). Note that the Attacker knows the Defender’s strategy and can observe its moves, and hence he can recognize this attack opportunity. Since the Defender needs at least 4 time units to return to \( \tau_2 \), the expected damage caused by this attack is at least 8.

The Attacker’s ability to anticipate future Defender’s moves can be decreased by randomization. Consider the simple strategy \( \sigma_b \) of Fig. 2b, where the Defender moves from \( v \) to \( \tau_1 \) and \( \tau_2 \) with probability \( p \) and \( 1 - p \), respectively. After reaching a target, the Defender returns to \( v \). The optimal value of \( p \) is \( \frac{1}{2} - \frac{\sqrt{3}}{2} \approx 0.3 \), and the expected damage of an optimal attack is then \( \approx 7.7 \). Note that the strategy \( \sigma_b \) is memoryless, i.e., its decisions depend just on the currently visited vertex.

At first glance, it is not clear whether the protection can be improved, because the probability \( p \) used by \( \sigma_b \) implements an optimal “balance” between visiting \( \tau_1 \) and \( \tau_2 \) determined by the weights of \( \tau_1 \) and \( \tau_2 \). However, consider the finite-memory strategy \( \sigma_c \) of Fig. 2c which is “almost deterministic” except for the moment when the Defender returns to \( v \) from \( \tau_2 \). Here, the strategy select the next edge uniformly at random. Then, the expected damage of an optimal attack is 6 (the best attack opportunity is to attack \( \tau_2 \) right after the robot starts moving form \( \tau_2 \) to \( v \). The expected time need to visit \( \tau_2 \) is then equal to 3, yielding the expected damage 6). Hence, protection is increased not only by randomization, but also by appropriate use of memory.

### 3 FINITE-MEMORY DEFENDER’S STRATEGIES

In this section, we prove that finite-memory Defender’s strategies can achieve the same limit protection value as general strategies.

Let \( G \) be a patrolling graph. A general Defender’s strategy for \( G \) (see Section 2.2.1) depends on the whole history of moves and cannot be finitely represented. A computationally feasible subclass are finite-memory (or regular) strategies [22, 23, 25] where the relevant information about the history is represented by finitely memory elements assigned to each vertex.

Formally, let mem: \( V \rightarrow \mathbb{N} \) be a function assigning to every vertex the number of memory elements. The set of augmented vertices is defined by \( \tilde{V} = \{(v, m): v \in V, 1 \leq m \leq \text{mem}(v)\} \). We use \( \tilde{v} \) to denote an augmented vertex of the form \((v, m)\) where \( m \leq \text{mem}(v) \).

A regular Defender’s strategy for \( G \) is a function \( \tilde{\sigma}: \tilde{V} \rightarrow \text{Dist}(\tilde{V}) \) where \( \sigma(v, m)(v', m') > 0 \) only if \( v \rightarrow v' \). We say that \( \sigma \) is unambiguous if for all \( v, v' \in V \) and \( m \leq \text{mem}(v) \) there is at most one \( m' \) such that \( \sigma(v, m)(v', m') > 0 \).

Intuitively, the Defender starts patrolling in a designated initial vertex \( v \) with initial memory element \( m \), and then traverses the vertices of \( G \) and updates the memory according to \( \sigma \). Hence, the current memory element represents some information about the history of visited vertices.

For every initial \( \tilde{v} \in \tilde{V} \), the strategy \( \sigma \) determines the probability space over the walks in the way described in Section 2.2.1. The only difference is that the probability of walk \( (v_1, \ldots, v_n) \) where \( v_1 = \tilde{v} \) is defined as \( \sum_{\tilde{v}_2, \ldots, \tilde{v}_n} \prod_{i=1}^{n-1} \sigma(\tilde{v}_i)(\tilde{v}_{i+1}) \). Here, \( \tilde{v}_1 = \tilde{v} \) is the initial augmented vertex. Hence, the notion of protection defined in Section 2.3 is applicable also to regular strategies (where \( \text{min}_{\sigma} \) is replaced with \( \text{min}_{\sigma} \) in (1)).

An important question is whether regular strategies can achieve the same limit protection value as general strategies. The answer is positive, and it is proven in two steps. First, we show that there exists an optimal Defender’s strategy \( \gamma \) satisfying \( \text{Val}(\gamma) = \text{Val} \) (see Section 2.3). Then, for arbitrarily small \( \epsilon > 0 \), we prove the existence of a regular strategy \( \sigma \) such that \( \text{Val}(\sigma) \leq \text{Val}(\gamma) + \epsilon \).

**Theorem 3.1.** For every patrolling graph, there exists a Defender’s strategy \( \gamma \) such that \( \text{Val}(\gamma) = \text{Val} \).

**Proof.** By the definition of \( \text{Val} \), there exist a vertex \( v \) and an infinite sequence \( \Gamma = \gamma_1, \gamma_2, \ldots \) of Defender’s strategies such that \( \text{Val}(\gamma_i) = \sup_{\sigma} \mathbb{E}^v \sigma[D^i]^\infty \) for all \( i \geq 1 \), and the infinite sequence \( \text{Val}(\gamma_1), \text{Val}(\gamma_2), \ldots \) converges to \( \text{Val} \).

Let \( \text{Histories} = h_1, h_2, \ldots \) be a sequence where every history occurs exactly once (without any special ordering). Let \( \Gamma_0 = \Gamma \). For every \( i \geq 1 \), we inductively define an infinite sequence of strategies \( \Gamma_i \) and a probability distribution \( \gamma_i(h_i) \) over \( V \), assuming that \( \Gamma_{i-1} \) has already been defined. Let \( t \) be the last vertex of \( h_i \), and let

![Figure 1: We compare two strategies (b) and (c) on graph (a). Strategy (b) patrols the targets in a deterministic loop, while strategy (c) applies randomization. Both have the same expected time lag between two consecutive visits to a target but differ in the protection significantly.](image1.png)

![Figure 2: On graph (a), a memoryless randomized strategy (b) is outperformed by a randomized strategy (c) with finite memory.](image2.png)
\{u_1, \ldots, u_k\} be the set of all immediate successors of \(t\) in \(G\) (i.e., \(t \rightarrow u_j \) for all \(j \leq k\)). Since every bounded infinite sequence of real numbers contains an infinite convergent subsequence, there exists an infinite subsequence \(\Gamma_t = \varrho_1, \varrho_2, \ldots \) of \(\Gamma_{t-1}\) such that the sequence 
\[ \varrho_1(h_t(u_i)), \, \varrho_2(h_t(u_i)), \ldots \]
is convergent for every \(j \leq k\). We put 
\[ \gamma(h_t(u_i)) = \lim_{t \to \infty} \varrho_t(h_t(u_i)). \]

It is easy to check that \(\sum_{j=1}^k \gamma(h_t(u_i)) = 1\), i.e., \(\gamma(h_t)\) is indeed a distribution on \(V\). Hence, the function \(\gamma\) is a Defender’s strategy, and we show that \(\text{Val}(\gamma) = \text{Val}\).

For the sake of contradiction, suppose \(\sup_{\pi} E^{V,\pi}[D^\pi] - \text{Val} = \delta > 0\). Then, \(\epsilon = \delta/4\). Then, there exists an Attacker’s strategy \(\pi^*\) such that
\[ E^{V,\pi}[D^\pi] \geq \text{Val} + \delta - \epsilon \tag{3} \]
Let \(A^t(\Omega) \subseteq \Omega\) be the set of all observations \(\omega\) such that \(\pi^*(\omega) = \text{att}_{\tau}\) for some \(\tau \in T\). For every \(\omega = o_1, \ldots, o_n, o_n = o_{n+1} \in A^t(\Omega)\), let \(w(\omega)\) be the set of all walks starting with \(o_1, \ldots, o_{n+1}\). Observe that if \(o, o' \in A^t(\Omega)\) and \(o \neq o'\), then \(w(o) \cap w(o') = \emptyset\). Furthermore, for every \(w \in W \setminus \bigcup_{\omega \in A^t(\Omega)} w(\omega),\) we have \(D^\pi_\omega(w) = 0\). Hence, we obtain
\[ E^{V,\pi}[D^\pi] = \sum_{\omega \in A^t(\Omega)} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}[D^\pi | w(\omega)] \tag{4} \]
where \(E^{V,\pi}[D^\pi | w(\omega)]\) is the conditional expected value of \(D^\pi\) under the condition that a walk starts with \(\omega\). (Note that the conditional expectation is undefined when \(E^{V,\pi}(w(\omega)) = 0\). In that case, \(0 \cdot \text{undefined}\) is interpreted as 0.) Hence, there exists a finite set of observations \(O \subseteq A^t(\Omega)\) such that
\[ E^{V,\pi}[D^\pi] \leq \epsilon + \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}[D^\pi | w(\omega)] \tag{5} \]
For each \(\omega = o_1, \ldots, o_n, o_n = o_{n+1} \in O\), let \(\tau_0\) be the target attacked by \(\pi\) after observing \(\omega\), and let \(H(\omega)\) be the set of all histories initiated in \(o_{n+1}\) such that \(\tau_0\) is the last vertex of \(h\). We use \(\omega \circ h\) to denote the history \(o_1, \ldots, o_n, h\). We have that \(E^{V,\pi}[D^\pi | w(\omega)] = \text{Val}\) is equal to
\[ \sum_{h \in H(\omega)} E^{V,\pi}(w(\omega \circ h) | w(\omega)) \cdot \text{tm}(\omega) \cdot \text{am}(\tau_0) \tag{6} \]
where \(E^{V,\pi}(w(\omega \circ h) | w(\omega))\) is the conditional probability of performing a walk starting with \(\omega \circ h\) under the condition that a walk starting with \(\omega\) is performed. Clearly, there exists a finite \(H(\omega) \subseteq H(\omega)\) such that the sum (6) decreases at most by \(\epsilon\) when \(h\) ranges over \(H(\omega)\) instead of \(\hat{H}(\omega)\). For short, we use \(E^{V,\pi}(\omega)\) to denote the sum
\[ \sum_{h \in H(\omega)} E^{V,\pi}(w(\omega \circ h) | w(\omega)) \cdot \text{tm}(\omega) \cdot \text{am}(\tau_0) \tag{7} \]
Hence,
\[ E^{V,\pi}[D^\pi | w(\omega)] \leq \epsilon + E^{V,\pi}(\omega) \tag{8} \]
By combining (5) and (8), we obtain
\[ E^{V,\pi}[D^\pi] \leq 2\epsilon + \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}(\omega) \tag{9} \]
Since \(\epsilon = \delta/4\), this implies
\[ \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}(\omega) \geq \text{Val} + \epsilon \tag{10} \]
Let \(H = \bigcup_{\omega \in O} H(\omega)\). Since \(H\) is finite, there exists an index \(\ell\) such that all elements of \(H\) appear among the first \(\ell\) elements of \(\text{Histories}\). Consider the sequence \(\Gamma_{\ell} = \varrho_1, \varrho_2, \ldots\) and observe that, for all \(i \geq 1\),
\[ \text{Val}(\varrho_i) = \sup_{\pi} E^{V,\pi}[D^\pi] \geq \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}(\omega) \tag{11} \]
Since the sequence of distributions \(\varrho_1(h), \varrho_2(h), \ldots\) converges to \(\gamma(h)\) for every \(h \in H\), we also obtain that
\[ \lim_{t \to \infty} \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}(\omega) = \sum_{\omega \in O} E^{V,\pi}(w(\omega)) \cdot E^{V,\pi}(\omega) \tag{12} \]
By combining (10), (11), and (22), we obtain \(\text{Val}(\varrho_j) \geq \text{Val} + \epsilon/2\) for all sufficiently large \(j\). This means that the sequence \(\text{Val}(\varrho_j), \text{Val}(\varrho_2), \ldots\) does not converge to \(\text{Val}\), and we have a contradiction. \(\square\)

**Theorem 3.2.** Let \(G\) be a patrolling graph, and let \(\text{Reg}\) be the class of all regular strategies for \(G\). Then, \(\inf_{\sigma \in \text{Reg}} \text{Val}(\sigma) = \text{Val}\).

**Proof.** By Theorem 3.1, there exists a Defender’s strategy \(\gamma\) such that \(\text{Val}(\gamma) = \text{Val}\). We show that for every \(\epsilon > 0\), there exist sufficiently large \(d, \ell \in \mathbb{N}\) such that \(\text{Val}(\sigma_{d,\ell}) \leq \text{Val}(\gamma) + \epsilon\), where \(\sigma_{d,\ell}\) is a regular strategy obtained by \(d\)-discretization and \(\ell\)-folding of \(\gamma\).

A \(d\)-discretization of \(\gamma\) is a strategy \(\gamma_d\) for which all \(h \in H\) and \(v \in V\), the following conditions are satisfied:
\[ \gamma_d(h)(v) = k/d \text{ for some } k \in \{0, \ldots, d\}; \]
\[ \gamma_d(h)(v) = 0 \text{ iff } \gamma(h)(v) = 0; \]
\[ |\gamma_d(h)(v) - \gamma(h)(v)| \leq |V|/d. \]
Observe that a \(d\)-discretization of \(\gamma\) exists for every \(d \geq |V|\).

The regular strategy \(\sigma_{d,\ell}\) is obtained by \(\ell\)-folding the strategy \(\gamma_d\) constructed for a sufficiently large \(d\). We can view \(\gamma_d\) as an infinite tree \(T\) where the nodes are histories and \(h \sim u\) iff \(\gamma(h)(u) = x\). Furthermore, we label each node \(h\) of \(T\) with the last vertex of \(h\). Since the edge probabilities range over finitely many values, the tree \(T\) contains only finitely many subtrees of height \(d\) up to isomorphism preserving both node and edge labels. For a given history \(h = v_0, \ldots, v_n\), let \(f_h, s_h \in \mathbb{N}\) be the lexicographically smallest pair of indexes such that \(f_h < s_h\), both \(f_h\) and \(s_h\) are integer multiples of \(d\), and the subtrees of height \(d\) rooted by \(v_{f_h}\) and \(s_h\) are isomorphic. If no such \(f_h, s_h\) exist, we say that \(h\) does not contain a folding pair. Note that there exists a constant \(c_f\) independent of \(h\) such that every history \(h\) of length at least \(c_f\) contains a folding pair \(f_h, s_h\) with both components bounded by \(c_f\). We define a strategy \(\gamma_{d,\ell}\) as follows:
\[ \gamma_{d,\ell}(h) = \gamma_d(h) \text{ for every history } h \text{ without a folding pair.} \]
\( y_{d,h}(t) = y_{d,t}(h') \) for every history \( h \) with a folding pair \( f_{i-1}, f_{i} \), where the \( h' \) is obtained from \( h = v_0, \ldots, v_n \) by deleting the subpath \( v_{i-1} \cdots v_{i-1} \).

Note that \( y_{d,h} \) can be equivalently represented as a \textit{finite-memory} strategy \( y_{d,t} \) where the memory elements correspond to the (finitely many) histories without a folding pair.

It remains to show that for every \( \epsilon > 0 \) there are sufficiently large \( d, t \) such that \( \text{Val}(y_{d,h}) \leq \text{Val}(\gamma) + \epsilon \), which proves our theorem.

For the rest of this proof, we fix \( \epsilon > 0 \). Furthermore, we fix \( k \in \mathbb{N} \) satisfying

\[
\sum_{i=2}^{\infty} \frac{1}{2^{k(i-1)}} \cdot i \cdot k \cdot 2 Val \cdot t m_{\text{max}} \cdot \alpha_{\text{max}} \leq \epsilon/4 \tag{18}
\]

Here \( t m_{\text{max}} = \max\{m(e) \mid e \in E\} \) and \( \alpha_{\text{max}} = \max\{a(\tau) \mid \tau \in T\} \). Note that such a \( k \) exists because the above sum converges to 0 as \( k \to \infty \).

For every \( i \in \mathbb{N} \), let \( I_i \) be the set of all integers \( j \) satisfying

\[
(i-1) \cdot k \cdot 2 Val \leq j < i \cdot k \cdot 2 Val
\]

We have that

\[
\mathbb{E}[h, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j] \cdot \mathbb{P}[h, u]_{R_{r_j} \in I_j} \tag{19}
\]

Observe

\[
\mathbb{P}[h, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j] \leq \frac{i \cdot k \cdot 2 Val \cdot t m_{\text{max}} \cdot \alpha_{\text{max}}}{2^{k(i-1)}} \tag{20}
\]

Inequality (20) is trivial, and (21) follows from (17). Using (18), we obtain

\[
\sum_{i=2}^{\infty} \frac{1}{2^{k(i-1)}} \cdot i \cdot k \cdot 2 Val \cdot t m_{\text{max}} \cdot \alpha_{\text{max}} \leq \epsilon/4 \tag{22}
\]

Consider a strategy \( y_d \) where \( d \geq |V| \). Then every \( \gamma \)-eligible history is \( y_d \)-eligible, and vice-versa. Furthermore, for every \( \delta > 0 \), there exists a sufficiently large \( d \) such that, for all \( h, \tau, r, \) and \( i \),

\[
\mathbb{E}[h, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j] \leq \mathbb{P}[h, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j] + \delta}
\]

Consequently, we can fix a sufficiently large \( d \) such that the values of

\[
\sum_{i=2}^{\infty} \frac{1}{2^{k(i-1)}} \cdot i \cdot k \cdot 2 Val \cdot t m_{\text{max}} \cdot \alpha_{\text{max}} \leq \epsilon/4
\]

Now consider a strategy \( y_{d,t} \) where \( t = k \cdot 2 Val \), and let \( h \) be a \( y_{d,t} \)-eligible history. Then \( \mathbb{E}[y_{d,t}, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j]} \) is equal to

\[
\sum_{i=1}^{\infty} \frac{1}{2^{k(i-1)}} \cdot i \cdot k \cdot 2 Val \cdot t m_{\text{max}} \cdot \alpha_{\text{max}} \leq \epsilon/2 \tag{23}
\]

By definition of \( \delta \)-folding, for every \( i \geq 1 \) there is a \( y_{d,t} \)-eligible history \( h' \) without a folding pair such that

\[
\mathbb{E}[y_{d,t}, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j] = \mathbb{P}[y_{d,t}, u]_{\mathcal{D}[\mathcal{P}[r_j] | R_{r_j} \in I_j} \tag{24}
\]

Furthermore,

\[
\mathbb{P}[y_{d,t}, u]_{R_{r_j} \in I_j] \leq (1 - \xi)^k (i-1) \tag{25}
\]
where $\xi$ is the maximal $\sup_{v \in h'} u_r | R_e \in I_1 |$ such that $h'$ is a $\gamma_d$-eligible history $h'$ without a folding pair.

Due to (24) and (25), the bounds of (23) remain valid when $\gamma_d$ is replaced with $\gamma_{df}$ and $h$ is a $\gamma_{df}$-eligible history. Hence, 

$$\mathbb{E}^{\gamma_{df}[h \cup | D^\pi[r]]} \leq \text{Val} + \epsilon$$ (26)

Since (26) holds for every $\gamma_{df}$-eligible $h$ and every target $\tau$, we obtain $\text{Val}(\gamma_{df}) \leq \text{Val} + \epsilon$. □

4 STRATEGY SYNTHESIS ALGORITHM

In this section, we design an efficient algorithm synthesizing a regular Defender’s strategy for a given patrolling graph capable of balancing the trade-off between maximizing $\text{Val}(\sigma)$ and minimizing the tail probabilities $P_c(\text{Val}(\sigma))$.

4.1 Computing $\text{Val}(\sigma)$

First, we show how to compute $\text{Val}(\sigma)$ for a given regular strategy $\sigma$. Let $G$ be a patrolling graph and $\sigma$ a regular strategy for $G$. Let $\bar{E}$ be the set of all $(\bar{G}, \bar{V}) \in \bar{V} \times \bar{V}$ such that $(\bar{G}(\bar{V})) \geq 0$, i.e., $\bar{E}$ is the set of augmented edges used by $\sigma$. For every target $\tau$, let $\pi[\tau]$ be the Attacker strategy where for all $u, v \in \bar{V}$ we have that $\pi[\tau](u, u \rightarrow v) = \text{attack}_\tau$, i.e., $\pi[\tau]$ attacks $\tau$ immediately after the Defender starts its walk.

For every $\tilde{e} = (\bar{G}, \bar{V}) \in \bar{E}$ and $\tau \in T$, let $L_{\tau, \tilde{e}}$ be the expected damage caused by an attack at $\tau$ scheduled right after the Defender starts traversing $\tilde{e}$, i.e.,

$$L_{\tau, \tilde{e}} = \mathbb{E}^{\sigma, \tilde{e}}[D^\pi[r] | \text{ walk} (\tilde{e})].$$ (27)

Hence, $L_{\tau, \tilde{e}}$ is the conditional expected value of $D^\pi[r]$ under the condition that the Defender’s walk starts by traversing $\tilde{e}$.

Consider the directed graph $\bar{G} = (\bar{V}, \bar{E})$, and let $\bar{B}$ denote the set of all bottom strongly connected components of $\bar{G}$. Let

$$\mathcal{L}(\sigma) = \min_{B \in \bar{B}} \max_{\tau \in T} L_{\tau, \tilde{e}}$$ (28)

where $\mathcal{E}(B) = \bar{E} \cap (B \times B)$ is the set of augmented edges in the component $B$ used by $\sigma$. We have the following:

Theorem 4.1. Let $\sigma$ be a regular strategy for a patrolling graph $G$. Then $\text{Val} (\sigma) \leq \mathcal{L}(\sigma)$. If $\sigma$ is unambiguous, then $\text{Val}_u (\sigma) = \mathcal{L}(\sigma)$.

Proof. For purposes of this proof, we need to introduce several notions. An augmented walk is an infinite sequence $v_1, v_2, \ldots$ such that $v_i \rightarrow v_{i+1}$ for all $i$. The set of all augmented walks is denoted $W$. An augmented history is a non-empty finite prefix of an augmented walk. The set of all augmented histories is denoted by $H$, and for $h \in H$, $\text{walk}(h)$ denotes the set of all augmented walks starting with $h$.

Furthermore, for every regular strategy $\sigma$ and every initial $\bar{o} \in \bar{V}$, we define the probability space $(W, F, \mathbb{P}^{\sigma, \bar{o}})$ over the augmented walks in the expected way.

For an arbitrary walk $w = v_1, v_2, \ldots$, let $\text{Aug}(w)$ denote the set of all augmented walks of the form $u_1, u_2, \ldots$. Observe that for every measurable event $F$ in $\mathcal{F}$, writing $\mathcal{F} \triangleq \bigcup_{w \in W} \text{Aug}(w)$, we have $\mathcal{F} \in \mathcal{F}$ and $\mathbb{P}^{\sigma, \bar{o}}(\mathcal{F}) = \mathbb{P}^{\sigma, \bar{o}}(F)$. Hence, to simplify our notation, we write $\mathcal{F}$ instead of $\mathcal{F}$.

The extension of the random variable $D^\pi$ to augmented walks is straightforward.

Let $\sigma$ be a regular Defender’s strategy. We prove $\text{Val}(\sigma) \leq \mathcal{L}(\sigma)$. Recall

$$\text{Val}(\sigma) = \min \sup_{\bar{o} \in \bar{V}} \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]]$$

$$\mathcal{L}(\sigma) = \min \max_{B \in \bar{B}} \min \max_{\tau \in T} L_{\tau, \tilde{e}}$$

Let $B' \in \bar{B}$ be a bottom strongly connected component achieving the above minimum. We show that

$$\min \sup_{\bar{o} \in \bar{V}} \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]] \leq \max_{\tau \in T} \min \max_{\tilde{e} \in E(B')} L_{\tau, \tilde{e}}.$$

Denoting the right-hand side by $M$, we show that there is $\bar{e} \in \bar{V}$ such that $\sup_{\bar{o}} \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]] \leq M$. Choose an arbitrary $\bar{e} \in B'$ and assume, for the sake of contradiction, that $\sup_{\bar{o}} \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]] > M$.

Hence, there is an Attacker’s strategy $\pi$ such that $\mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]] > M$.

Now we decompose the expectation $\mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]]$ according to augmented observations, i.e., sequences $\bar{o} = v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$, where $v_1, \ldots, v_n$ is a path in $G$. Given $\bar{o} = v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$, we use $o$ to denote the “unaugmented” observation $v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$. Let $\bar{O}$ be the set of all augmented observation, and let $\text{Att}(\bar{O}) \subseteq \bar{O}$ be the set of all $\bar{o}$ where $\pi(\bar{o}) \neq \pi(\bar{o})$. For every $\bar{o} = v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$ in $\text{Att}(\bar{O})$, let $\text{walk}(\bar{o})$ be the set of all augmented walks starting with $v_1, \ldots, v_{n+1}$. Note that if $\bar{o} = v_1, \ldots, v_n, v_n \rightarrow v_{n+1}$ in $\text{Att}(\bar{O})$, let $\text{walk}(\bar{o})$ be the set of all augmented walks starting with $v_1, \ldots, v_{n+1}$. Furthermore, for every $\bar{o} \in \bar{O}$, we have that $\mathcal{D}^\pi[\bar{o}] = 0$. Hence, we obtain

$$\mathbb{E}^{\sigma, \bar{o}}[D^\pi[r]] = \sum_{\bar{o} \in \text{Att}(\bar{O})} \mathbb{P}^{\sigma, \bar{o}}[\text{walk}(\bar{o})] \cdot \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r] \mid \text{walk}(\bar{o})].$$

Since

$$\sum_{\bar{o} \in \text{Att}(\bar{O})} \mathbb{P}^{\sigma, \bar{o}}[\text{walk}(\bar{o})] \leq 1,$$

there is $\bar{o} = v_1, \ldots, v_n, v_n \rightarrow v_{n+1} \in \text{Att}(\bar{O})$ such that

$$\mathbb{P}^{\sigma, \bar{o}}[\text{walk}(\bar{o})] > 0 \quad \text{and} \quad \mathbb{E}^{\sigma, \bar{o}}[D^\pi[r] \mid \text{walk}(\bar{o})] > M.$$
For every observation \( o \in \Omega \) s.t. \( P^\sigma,\tilde{d}(\text{walk}(o)) > 0 \), there is exactly one augmented observation of the form \( \tilde{d} \) such that \( P^\sigma,\tilde{d}(\text{walk}(\tilde{d})) > 0 \) (this follows by a trivial induction on the length of \( o \)). In the rest of this proof, for every \( o \in \Omega \), the symbol \( \tilde{d} \) denotes the unique augmented observation satisfying the above. Now, for each \( o \in \Omega \), we define \( \pi(o) \) as follows: Let \( \tilde{d} = \tilde{d}_1, \ldots, \tilde{d}_n, \tilde{d}_n \rightarrow \tilde{d}_{n+1} \). If none of the augmented edges \( (\tilde{d}_i,\tilde{d}_{i+1}) \) for \( 1 \leq i < n \) is in \( \mathcal{E} \) and the last augmented edge \( \tilde{e} = (\tilde{d}_n,\tilde{d}_{n+1}) \) does appear in \( \mathcal{E} \), we put \( \pi(o) = \text{attack}_{\tilde{d}_n}(\tilde{e}) \). Otherwise, we put \( \pi(o) = \text{wait} \). For every \( \tilde{e} \in \mathcal{E} \), let \( \text{Att}(\tilde{e}) \) denote the union of \( \text{walk}(\tilde{d}) \) over all \( o \in \Omega \) such that \( \pi(o) \neq \text{wait} \) and \( \tilde{d} \) ends with \( \tilde{e} \). Since the Attacker may attack only once along any (augmented) walk, we have \( \text{Att}(\tilde{e}) \cap \text{Att}(\tilde{e}') = \emptyset \) for all \( \tilde{e}, \tilde{e}' \in \mathcal{E} \) such that \( \tilde{e} \neq \tilde{e}' \). Since \( \mathcal{D}^\sigma \) is non-negative, we obtain

\[
E^{\sigma,\tilde{d}}[\mathcal{D}^\sigma] \geq \sum_{\tilde{e} \in \mathcal{E}} P^{\sigma,\tilde{d}}(\text{Att}(\tilde{e})) \cdot E^{\sigma,\tilde{d}}[\mathcal{D}^\sigma | \text{Att}(\tilde{e})]. \tag{31}
\]

Since \( \sigma \) is regular and, for every \( \tilde{e} \in \mathcal{E}, \pi \) always attacks the same target \( \tau(\tilde{e}) \) when the Defender starts traversing \( \tilde{e} \) (regardless of the previous history), we have

\[
E^{\sigma,\tilde{d}}[\mathcal{D}^\sigma | \text{Att}(\tilde{e})] = L_{\tau(\tilde{e}),\tilde{e}} \geq L(\sigma),
\]

where the inequality follows from (30). Substituting into (31) yields

\[
E^{\sigma,\tilde{d}}[\mathcal{D}^\sigma] \geq L(\sigma) \sum_{\tilde{e} \in \mathcal{E}} P^{\sigma,\tilde{d}}(\text{Att}(\tilde{e})).
\]

The desired inequality \( E^{\sigma,\tilde{d}}[\mathcal{D}^\sigma] \geq L(\sigma) \) follows from the fact that the above sum is equal to 1 (this follows by applying basic results of finite Markov chain theory; the Defender almost surely visits some bottom strongly connected component \( B \in \mathcal{B} \) and there it almost surely traverses every edge infinitely often. In particular, the Defender almost surely visits the unique edge \( \tilde{e} \in (E \cap E(B)) \).

Our proof of Theorem 4.1 reveals that the Attacker can cause the expected damage equal to \( L(\sigma) \) if it can observe the memory updates performed by the Defender. If \( \sigma \) is unambiguous, then the Attacker can determine the memory updates just by observing the history of Defender’s moves. However, if the memory updates are randomized, the Attacker needs to access the Defender’s internal data structures during a patrolling walk. Depending on a setup, this may or may not be possible. By the worst-case paradigm of adversarial patrolling, \( L(\sigma) \) is more appropriate than \( \text{Val}(\sigma) \) for measuring the protection achieved by \( \sigma \). As we shall see, our strategy synthesis algorithm typically outputs unambiguous regular strategies where \( L(\sigma) = \text{Val}(\sigma) \). Hence, it does not really matter whether \( L(\sigma) \) is understood as the protection achieved by \( \sigma \) or just a bound on this protection.

Fix \( B \in \mathcal{B} \) and \( \tau \in \mathcal{T} \). If \( B \) does not contain any augmented vertex of the form \( \tilde{e} \), then \( L_{\tau,\tilde{e}} = \infty \) for all \( \tilde{e} \in E(B) \). Otherwise, to every \( \tilde{e} = (\tilde{u},\tilde{v}) \in E(B) \), we associate a variable \( X_{\tilde{e}} \), and create a system of linear equations

\[
X_{\tilde{e}} = tm(u,v) + \begin{cases} 0 & \text{if } \nu = \tau, \\ \sum_{\tilde{\nu} \rightarrow \tilde{\nu}} \sigma(\tilde{\nu})(\tilde{\nu}) \cdot X_{\text{Att}(\tilde{\nu})} & \text{otherwise} \end{cases} \tag{32}
\]

over \( \tilde{e} \in E(B) \). By a straightforward generalization of [28, Theorem 1.3.5], system (32) has a unique solution, equal to \( (L_{\tau,\tilde{e}})_{\tilde{e} \in E(B)} \).

Observe that for all augmented edges \( \tilde{e}, \tilde{g} \in E(B) \) leading to the same augmented vertex (say \( \tilde{e} = (\tilde{u},\tilde{v}) \) and \( \tilde{g} = (\tilde{w},\tilde{v}) \)), we have \( L_{\tau,\tilde{e}} = L_{\tau,\tilde{e}} - tm(u,v) + tm(w,v) \). Therefore, we may reduce the number of variables as well as equations of the system from \( |E(B)| \) to \(|B| \). Indeed, to every \( \tilde{e} \in B \), we assign a variable \( Y_{\tilde{e}} \), and construct a system of linear equations

\[
Y_{\tilde{e}} = \begin{cases} 0 & \text{if } \nu = \tau, \\ \sum_{\tilde{\nu} \rightarrow \tilde{\nu}} \sigma(\tilde{\nu})(\tilde{\nu}) \cdot (tm(u,v) + Y_{\tilde{\nu}}) & \text{otherwise}. \end{cases} \tag{33}
\]

Then, for every \( \tilde{e} = (\tilde{u},\tilde{v}) \in E(B) \), we have \( L_{\tau,\tilde{e}} = tm(u,v) + y_{\tilde{e}} \), where \((y_{\tilde{e}})_{\tilde{e} \in B} \) is the unique solution of the system (33).

### 4.2 Optimization Scheme

Our strategy synthesis algorithm is based on interpreting \( L \) as a piecewise differentiable function and applying methods of differentiable programming. We start from a random strategy \( \sigma \), repeatedly compute \( L(\sigma) \) and update the strategy against the direction of its gradient.

The optimization algorithm is described in Algo. 1. On forward pass, strategies are produced from real-valued coefficients by a Softmax function that outputs probability distributions. For every target \( \tau \), we solve the system (33) to obtain a damage \( (L_{\tau,\tilde{e}})_{\tau \in \mathcal{T}, \tilde{e} \in E(B)} \). Then, instead of hard maximum in equation (28), we optimize a loss function defined by

\[
\text{loss} = \sum_{\tau,\tilde{e}} \Phi_\epsilon(L_{\tau,\tilde{e}})^2, \tag{34}
\]

where \( \Phi_\epsilon(t) = 0 \) for \( t \in [0,m-\epsilon m) \) and \( \Phi_\epsilon(t) = 1 + (t - m) / \epsilon m \) for \( t \in [m-\epsilon m, m) \), in which \( m = \sum \) is the hard maximum, the bar denotes the stop-operator, and \( \epsilon \in (0,1) \) is a hyperparameter. Minimizing loss instead of \( L \) leads to a more efficient gradient propagation. On top of this, we enforce the model to prefer deterministic strategies over randomized by adding an average entropy of strategies’ probability distributions with a factor \( \beta \).

On backward pass, the loss gradient is computed using the automatic differentiation, we add decaying Gaussian noise and update the coefficients using Adam optimizer [21].

As Softmax never produces probability distributions containing zeros, we cut the outputs at a certain threshold (called rounding threshold) to allow endpoint values on evaluation. Note that edges with zero probabilities are excluded from \( E(B) \) which is crucial for equation (33).

### 5 EXPERIMENTS

We experimentally evaluate strategy synthesis algorithm on series of synthetical graphs with increasing sizes. We perform two sets of tests. The first analyzes runtimes while the second one focuses on the achieved protection values.

The experiments were performed on a desktop machine with Ubuntu 20.04 LTS running on Intel® Core™ i7-8700 Processor (6 cores, 12 threads) with 32GB RAM.

#### 5.1 Runtime Analysis

We generate synthetic graphs with \( n = 10, 20, \ldots, 100 \) vertices. To obtain random but similarly structured graphs, we start with a grid of size \( n \times n \) and choose its \( n \) nodes as the vertices of our patrolling graph, half of them being targets. All vertices are equipped with 6 memory elements. The travel time between vertices is set to the number of edges on the shortest path in the original grid. In the
Algorithm 1 Strategy optimization

coefficients ← Init()
for step ∈ steps do
    // Forward pass
    strategy ← Softmax(coefficients)
    // solving linear system (33)
    damage ← Solve(strategy)
    loss ← Loss(damage) + 𝛽 · Entropy(strategy)
    // Backward pass
    coefficients.grad ← Gradient(loss)
    // automatic differentiation
    coefficients.grad += Noise(step)
    // Adam optimizer’s step
    coefficients += Step(coefficients.grad, step)
    // Strategy evaluation
    strategy ← Cutoff(Softmax(coefficients))
    damage ← Solve(strategy)
    𝐿 ← Max(damage)
    Save 𝐿, strategy
return strategy with the smallest 𝐿

final patrolling graph, we omit those edges that have an alternative connection of at most the same length visiting another vertex.

For each 𝑛, we generate 10 graphs of 𝑛 vertices and run 10 optimization trials with 100 steps for each graph. In Fig. 3, we report statistics of average step-times in seconds aggregated by 𝑛. Note than even considerably large graphs are processed in units of seconds, which confirms the applicability of our algorithm to dynamically changing environments (see Section 1).

Recall that one optimization step consists of a forward pass (damage and loss of the current strategy), backward pass (gradient), and one more test evaluation. For hyperparameters, we set 𝜖 = 0.3, 𝛽 = 0.2, learning rate = 0.5, cutoff threshold = 0.1, and rounding threshold = 0.001 (for a deeper explanation, see Section 4.2).

Figure 3: Runtime analysis of the strategy synthesis algorithm (in seconds).

Fig. 4 shows the convergence of values during the optimization process. Here we fix one graph for each number of vertices and run 10 trials each with 120 steps. The colors are assigned to individual graphs, the areas show ranges of the obtained values during the optimization process. The solid lines highlight the minimal values.

5.2 Patrolling Airport Gates

One typical application of patrolling is security patrol at airports (see Section 1). Airport buildings have a specific tree structure of terminals with symmetric gates with a central node connecting the terminals. A terminal typically consists of pairs of gates joined by halls. A patrolling graph for an airport with three terminals of 4, 2, and 6 gates is shown in Fig. 5.

![Figure 5: A patrolling graph for an airport with 3 terminals.](image1)

We generate a sequence of random airport graphs with 3 terminals and an increasing number of gates determined randomly. Hence, an airport graph with 𝑛 gates has 𝑛/2 halls and exactly one central node. The gates are targets and have exactly one memory element, while halls and the central node are non-target vertices with 4 memory elements each (thus, the strategy can “remember” the previously visited vertex). The costs of all targets are equal to 1, and all edges have the same traversal time 1.

Since the target costs are the same, we can estimate Val (the achievable protection) by the length of the shortest cycle visiting all targets, which is 2(|𝑉| − 1) where |𝑉| is the set of vertices. Note that automatic synthesis of a regular strategy with comparable protection is tricky—the synthesis algorithm must “discover” the
relevance of the previously visited vertex and design the memory updates accordingly.

We use the same hyperparameters as above. For each airport, we synthesized 30 strategies, each in 500 steps of iterations. In Fig. 6, we show the protection values of the synthesized strategies for increasing number of vertices normalized by the baseline $2(|V| - 1)$. For larger $|V|$, the optimization converges to locally optimal randomized strategies with worse protection than the deterministic-loop strategy. In particular, for $|V| = 91$, the protection achieved by the constructed strategy is about 33% worse than the baseline on average (with best found strategy loosing 20% above the baseline).

Figure 6: Normalized values of strategies synthesized for airport graphs with increasing number of vertices. The normalization baseline is the shortest cycle visiting all targets with length $2(|v| - 1)$ where $v$ stands for the number of vertices.

6 CONCLUSION

The outcomes show that high-quality Defender’s strategies are computed very quickly even for instances of considerable size. Hence, our algorithm can also be used to re-compute a Defender’s strategy dynamically when the patrolling scenario changes.

The problem encountered in our experiments is the existence of locally optimal randomized strategies where the optimization loop gets stuck. An interesting question is whether this problem can be overcome by tuning the parameters of gradient descent or by constructing the initial seeds in a more sophisticated way.

A natural continuation of our study is extending the presented results to scenarios with multiple Defenders and Attackers.

ACKNOWLEDGEMENTS

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0189.

Vít Musil is also supported from Operational Programme Research, Development and Education - Project Postdoc2MUNI (No. CZ.02.2.69/0.0/0.0/18_053/0016952).

Disclaimer. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government
