Adversarial Patrolling with Drones

David Klaška
Antonín Kučera
Vojtěch Řehák
{xklaska1,tony,rehak}@fi.muni.cz
Faculty of Informatics, Masaryk University
Brno, Czech Republic

ABSTRACT

We investigate *adversarial patrolling* where the Defender is an autonomous device with a limited energy resource (e.g., a drone). Every eligible Defender's policy must prevent draining the energy resource before arriving to a refill station, and this constraint substantially complicates the problem of computing an efficient Defender's policy. Furthermore, the existing infinite-horizon models assume Attackers with *unbounded patience* willing to wait arbitrarily long for a good attack opportunity. We show this assumption is *inappropriate* in the setting with drones because here the expected waiting time for an optimal attack opportunity can be extremely large. To overcome this problem, we introduce and justify a new concept of *impatient* Attacker, and design a *polynomial time* algorithm for computing a Defender's policy achieving protection close to the optimal value against an impatient Attacker. Since our algorithm can quickly evaluate the protection achievable for various topologies of refill stations, we can also optimize their displacement.

KEYWORDS

Single and multi-agent planning and scheduling; patrolling

ACM Reference Format:
David Klaška, Antonín Kučera, and Vojtěch Řehák. 2020. Adversarial Patrolling with Drones. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), Auckland, New Zealand, May 2020,* IFAAMAS, 9 pages.

1 INTRODUCTION

In general patrolling, a *Defender* moves among vulnerable *targets* and tries to discover ongoing attacks initiated by the *Attacker*. For every target v, completing an attack at v takes $d(v)$ time units, and an ongoing attack at v is discovered by the Defender only if she visits v in at most $d(v)$ time units since initiating the attack. A *policy* for the Defender specifies how she should move among the targets so that the overall protection is maximized (the policy can be *randomized*, i.e., the Defender can "flip a coin" where to go next). In *adversarial* patrolling based on *Stackelberg solution concept*, it is assumed that the Attacker not only knows the Defender's policy, but he can also observe the history of Defender's moves. Still, the Attacker cannot predict the way of resolving the Defender's randomized choice (the "coin flips"). This model is particularly appropriate in situations where the actual Attacker's capabilities and the amount of information revealed to him are *unknown* and robust policies are required.

In this paper, we investigate adversarial patrolling with robotic devices equipped with a bounded energy resource (hereafter called a *battery*) such as drones. Here, the Defender's policy must be *safe*, i.e., prevent draining the battery before arriving to a refill station. The "detours" to refill stations may negatively impact the overall protection, and the first question we study in our work is

What is the impact of battery capacity on achievable protection?

Our answer to this question is based on discovering a somewhat surprising dichotomy in Attacker's opportunities. Recall that in our setting, the Attacker knows the Defender's policy, observes her moves, and waits for the right moment (opportunity) to perform an attack maximizing his expected utility. Since the Defender's policy is randomized, the Attacker sees such an opportunity after a random time, and hence it makes sense to consider the *average* (expected) waiting time for an attack opportunity where the expected Attacker's utility exceeds a given threshold. Our first main result (Theorem 2.6) shows that there exists a *unique* threshold D with the following properties:

(A) For an arbitrarily large battery capacity c and an arbitrary safe Defender's policy, the Attacker will eventually see an attack opportunity with expected utility at least D, and the average waiting time for this opportunity is *linear* in c. In other words, no matter how large the battery capacity is and how cleverly the Defender moves from target to target, the Attacker can always get the expected utility at least D if he is willing to wait for time linear in c.

(B) The Defender has a safe policy σ such that the average waiting time for an attack opportunity *better* than D grows *exponentially* in the battery capacity c. Due to this exponential growth, even a small increase of battery capacity pushes the expected waiting time for an attack opportunity better than D to billions of years, which makes these attack opportunities infeasible.

This result clearly explains the actual impact of battery capacity—from certain point on, increasing the battery capacity does *not* improve the achievable protection, it just "pushes" the expected waiting time for an attack opportunity better than D to extremely large values. Hence, the threshold D corresponds to the maximal expected damage achievable by a realistic *impatient*[1] Attacker who is not willing to wait exponentially long. The corresponding maximal level of achievable *protection*[2] is denoted by *IVal*. Furthermore,

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 2020, Auckland, New Zealand. © 2020 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved. …$15.00

[1]Our notion of impatience should *not* be confused with a different concept of discounting the Attacker's utility also called "impatience" in [26].
[2]Technically, $IVal = \alpha_{\max} - D$, where $\alpha_{\max}$ is the maximal target importance (see Section 2). Alternatively, we could define $IVal = -D$; this would *not* change the

for a given lower bound τ on "infeasible" expected attack time, the underlying observations of Theorem 2.6 allow to determine a battery capacity $c[\tau]$ sufficiently large to push the expected waiting time for an attack opportunity better than D beyond τ. When τ is small, this can result in lowering the required battery capacity without compromising the achieved protection.

The previous results immediately motivate the second question considered in our paper:

Can we compute IVal and the Defender's policy σ of (B)?

Here, we face fundamental obstacles. The problem of computing an optimal or ε-optimal safe Defender's policy is NP-hard[3]. Actually, this hardness result holds even in the simpler setting without battery constraints (see Section 2). So, we cannot hope for an efficient algorithm computing (sub)optimal safe Defender's policies for all instances. Instead, we design a *policy-improvement* algorithm which repeatedly improves a given finite-memory Defender's policy by a modified gradient-descent method[4]. This improvement algorithm runs in *polynomial time* and can be quickly applied to many (hundreds or even thousands) initial policies chosen randomly or by some heuristics. By selecting the best of the obtained policies, we typically obtain a policy achieving reasonable protection. To evaluate these policies, we would need to know the actual value of *IVal*, which is hard to compute/approximate (see Section 2). Therefore, we design an algorithm computing an *upper bound* on *IVal*. Although this algorithm needs *exponential time* in general, it is sufficiently efficient to provide a good baseline for evaluating the computed policies.

The experiments presented in Section 4 show that the protection achieved by the computed policies typically stays close to *IVal*. Since our algorithm can quickly evaluate the protection achievable for various topologies of refill stations, it can also be used to optimize their displacement. We demonstrate this on a concrete example in Section 4.

Related work. Successful applications of drones stimulated intensive research in this area. The limitations implied by bounded battery capacity have been considered for various problems related to optimal trajectory planning for one or more drones, e.g., planning an optimal trajectory for visiting all targets under the battery constraint [13, 22], non-adversarial patrolling with battery-limited agents [24], an energy-bounded delivery by a drone [20], or the coverage problem for one or more vacuum cleaners with limited batteries [19, 23]. To the best of our knowledge, the problem of adversarial patrolling with drones has not yet been considered in previous works.

In security games, most of the existing works study either the problem of computing an optimal static allocation of available resources to the targets, or the problem of computing an optimal movement strategy for a mobile Defender. Security games with static allocation have been studied in, e.g., [12, 14, 16, 21, 25, 27, 28]. In patrolling games (where the Defenders are mobile), the focus was primarily on finding locally optimal strategies for robotic patrolling units either on restricted graphs such as circles [1, 2], or arbitrary graphs with weighted preference on the targets [3, 5]. Alternatively, the work focused on some novel aspects of the problem, such as variants with discounted rewards [26], moving targets [8, 11], multiple patrolling units [6], or movement of the Attacker on the graph [5], reaction to alarms [7, 10], and handling sequential attacks [18]. None of these results considers Defender(s) with a bounded energy resource.

Let us note that the previous works on adversarial patrolling with infinite horizon do not explicitly consider the amount of time the Attacker can wait for a good attack opportunity. In fact, it is mostly assumed that the Attacker's "patience" is unbounded, i.e., the Attacker is willing to wait arbitrarily long (in the context of patrolling *without* battery constraints, the expected waiting time for a good attack opportunity is *low*, and hence this simplifying assumption is harmless). One remarkable exception is [26], where later attacks are penalized by a discounting factor. This models the Attacker's preference to attack soon, even if the expected waiting time for an optimal attack opportunity is small. To some extent, this idea is orthogonal to our concept (after disregarding the attack opportunities with exponentially long waiting time, the Attacker may still consider earlier attacks as more valuable).

As we already noted, when patrolling with drones, the expected waiting time for the best attack opportunity (better than the threshold D introduced above) can be *very* large, and considering Attackers with unbounded patience leads to *unrealistic and misleading results*, as it is demonstrated by Examples 1 and 2 in Section 2.

Most of the existing algorithms solve the adversarial patrolling games using mathematical programming, e.g., [1, 3, 4, 9]. In our algorithm, we employ the polynomial-time procedure for regular strategy synthesis of [17] based on gradient descent. Recently, other learning approaches were applied to solving security games. One remarkable example is [15] where finite-horizon games are considered.

2 A FORMAL MODEL FOR PATROLLING WITH DRONES

In this section, we formalize the problem of patrolling with drones. Our model is similar to the standard model of patrolling games without battery constraints (see, e.g., [1, 4, 17]). Our notion of Defender's policy reflects the safety requirement, i.e., the battery must never be drained completely, and the Defender may use a different moving strategy for every battery capacity. The notion of Attacker's strategy is standard. The real difference is the way of evaluating the protection achieved by a given Defender's policy. Here, we need to find a robust way of disregarding Attacker's strategies with extremely long expected attack time, which leads to the notion of *impatient Attacker* and the value *IVal* justified by Theorem 2.6.

2.1 Patrolling graphs

The targets and refill locations are modeled as vertices in a directed graph, where the edges correspond to admissible moves of the drone. Some targets may also serve as refill locations. Although the

presented results, but negative numerical values of *IVal* would have less intuitive meaning from the Defender's point of view. We refer to Section 2 for a more detailed discussion.

[3]The NP hardness result for ε-optimal policy holds for $\varepsilon \leq 1/2n$, where n is the number of targets. The details are given at the end of Section 2.

[4]Our policy-improvement algorithm is based on combining the ideas behind the proof of Theorem 2.6 with the policy-improvement algorithm recently proposed in [17] for patrolling problems *without* battery constraints.

drone can move freely in principle, there can be restrictions such as no-fly zones or physical obstructions. Hence, the graph may not be fully connected in general. To every edge $v \to u$, we assign the number of time units needed to reach u from v. Although the edge relation is mostly symmetric, there may be exceptions due to traffic regulations, wind, etc., so we keep the graph directed. We assume that the amount of energy consumed by the drone when flying along an edge is proportional to the flying time. To each target v, we assign the amount of time units $d(v)$ needed to complete an attack at v. Since targets may have different importance, each target v is also assigned a non-negative number $\alpha(v)$, where higher value means higher importance.

Definition 2.1. A *patrolling graph* is a tuple $G = (V, T, R, \to, t, d, \alpha)$, where

- V is a non-empty set of *vertices*.
- $T \subseteq V$ and $R \subseteq V$ are non-empty sets of *targets* and *refill locations*, respectively, such that $T \cup R = V$. We do *not* require $T \cap R = \emptyset$, i.e., some targets can also serve as refill locations.
- $\to \subseteq V \times V$ is the set of *edges*.
- t is a function assigning to every edge the number of time units needed to traverse the edge.
- d is a function assigning to every target the number of time units needed to complete an attack at the target.
- α is a function assigning to every target a non-negative number specifying its *importance*.

2.2 Defender's policy

A *history* of G is a finite sequence $h = v_1, \ldots, v_n$ of vertices previously visited by the Defender (i.e., $n \geq 0$ and $v_i \to v_{i+1}$ for all $i < n$; note that the empty sequence of vertices, denoted by ε, is also a history). The set of all histories is denoted by $\mathcal{H}$. A *walk* in G is an infinite sequence of vertices w such that every finite prefix of w is a history of G.

A *moving strategy* for G is a function $\gamma : \mathcal{H} \to \Delta(V)$ where $\Delta(V)$ is the set of all probability distributions on V. We require that the distribution $\gamma(v_1, \ldots, v_n)$ assigns a positive probability only to immediate successors of v_n. In the special case when $n = 0$, i.e., $v_1, \ldots, v_n$ is the empty history ε, we have that $\gamma(\varepsilon)$ is an *initial* distribution over V.

Every moving strategy γ determines the probability space over the walks in the standard way[5]. We use $\mathcal{P}^\gamma$ to denote the associated probability measure. Let $c \in \mathbb{N}$ be a *battery capacity*. For every history $v_1, \ldots, v_n$ of G, let $b_1, \ldots, b_n$ be the corresponding sequence of *battery snapshots* defined inductively as follows:

- $b_1 = c$ (initially, the battery is fully charged).
- If $v_i \in R$, then $b_{i+1} = c - t(v_i, v_{i+1})$; otherwise, $b_{i+1} = b_i - t(v_i, v_{i+1})$.

The second item in the above definition says that traversing an edge $v \to u$ takes $t(v, u)$ units of energy (this reflects our assumption that the amount of consumed energy is proportional to the time needed to traverse the edge). When a refill location is visited, the battery is refilled to its full capacity. Without restrictions, we may assume that a refill takes zero time, because the time needed to perform a refill (i.e., replace the battery with a fully charged one) can be added to the time needed to traverse the edges.

A moving strategy γ is *c-eligible* if for every history $v_1, \ldots, v_n$ such that $\mathcal{P}^\gamma(v_1, \ldots, v_n) > 0$ we have that the corresponding sequence $b_1, \ldots, b_n$ of battery snapshots contains only non-negative numbers. We use ℓ_G to denote the least $c \in \mathbb{N}$ such that there exists a c-eligible moving strategy for G. Note that ℓ_G is easy to compute by employing standard graph algorithms.

Definition 2.2. Let G be a patrolling graph. A *Defender's policy* is a function σ assigning to every $c \geq \ell_G$ a c-eligible moving strategy σ_c.

2.3 Modeling the Attacker

In the existing adversarial patrolling models, the Attacker knows the Defender's policy and can observe Defender's moves. Furthermore, the Attacker can attack *at most once* during a play. This is no restriction, because defending a set of targets against multiple attacks is essentially the same optimization problem[6].

In our setting, the time is spent by flying along edges, not by staying in vertices. In the *worst case*, the Attacker can determine the next edge taken by the drone *immediately* after the drone leaves the currently visited vertex. Hence, the Attacker's decision is based not only on the sequence of vertices visited by the drone so far, but also on the edge taken next. Clearly, the Attacker cannot gain anything by delaying his attack until the drone arrives to the next vertex, i.e., we can safely assume an attack is initiated at the moment when the drone leaves the currently visited vertex. Furthermore, we assume the Attacker knows the battery capacity.

Definition 2.3. Let G be a patrolling graph. An *observation* is a finite sequence $o = v_1, \ldots, v_n, v_n{\to}u$, where $v_1, \ldots, v_n$ is a non-empty history and $v_n{\to}u$ is an edge. The set of all observations is denoted by Ω.

An *Attacker's strategy* for a patrolling graph G is a function π assigning to every battery capacity c a function

$$\pi_c : \Omega \to \{\mathit{wait}, \mathit{attack}_z \mid z \in T\}.$$

We require that if $\pi_c(v_1, \ldots, v_n, v_n{\to}u) = \mathit{attack}_z$ for some $z \in T$, then $\pi_c(v_1, \ldots, v_i, v_i{\to}v_{i+1}) = \mathit{wait}$ for all $1 \leq i < n$ (i.e., the Attacker can attack at most once).

2.4 Evaluating a Defender's policy

For the rest of this paragraph, we fix a patrolling graph G. Observe that for every Attacker's strategy π, every walk w in G, and every battery capacity c, the Attacker either attacks some vertex along w or not. In the first case, we say that his attack time is equal to the total time elapsed since initiating the walk until performing the attack; in the latter case, we say that his attack time is infinite. Recall that every Defender's policy determines a probability space over the walks, and then the attack time becomes a random variable

[5]More precisely, the probability space is $(\mathit{walk}, \mathcal{F}, \mathcal{P}^\gamma)$, where *walk* is the set of all walks, $\mathcal{F}$ is the (Borel) σ-algebra generated by all *basic cylinders*, i.e., all sets $\mathit{walk}(h)$, where h is a history, consisting of all walks starting with h. The probability measure $\mathcal{P}^\gamma$ is the unique probability measure satisfying $\mathcal{P}^\gamma(\mathit{walk}(h)) = \prod_{i=1}^{k-1} \gamma(v_1, \ldots, v_i)(v_{i+1})$, where $h = v_1, \ldots, v_k$.

[6]Even if the Attacker can perform another attack after completing the previous one, the best the Defender can do is to follow an optimal policy constructed for the single attack scenario. This is no longer true when the Defender has to spend some time responding the discovered attack [18].

with clearly defined mean (the expected value). A formal definition of attack time follows.

Definition 2.4. Let π be an Attacker's strategy. The π *attack time* for a battery capacity $c \geq \ell_G$ and a walk $w = v_1, v_2, \dots$ is defined by $Time^{\pi_c}(w) = \sum_{i=2}^{k} t(v_{i-1} \to v_i)$. Here, k is the least index s.t. $\pi_c(v_1, \dots, v_k, v_k \to v_{k+1}) = attack_z$ for some $z \in T$. If there is no such k, we put $Time^{\pi_c}(w) = \infty$.

Let π be an Attacker's strategy. We say that a target z is *penetrated* along a Defender's walk w for a battery capacity $c \geq \ell_G$ if

- $Time^{\pi_c}(w) = k < \infty$,
- $\pi_c(v_1, \dots, v_k, v_k \to v_{k+1}) = attack_z$,
- the target z is *not* among the vertices $v_{k+1}, \dots, v_{k+m}$, where $m \geq 0$ is the largest index such that

$$\sum_{i=1}^{m} t(v_{k+i-1} \to v_{k+i}) \leq d(z)\,.$$

Given a Defender's policy σ and an Attacker's strategy π, the probability of all walks w such that the target z is penetrated along w for c is denoted by $\mathcal{P}^{\sigma_c, \pi_c}(penetrate_z)$.

The *expected Attacker's utility for* c is defined by

$$\mathrm{EU}_A(\sigma_c, \pi_c) = \sum_{z \in T} \mathcal{P}^{\sigma_c, \pi_c}(penetrate_z) \cdot \alpha(z)$$

Intuitively, $\mathrm{EU}_A(\sigma_c, \pi_c)$ is the expected importance of a successfully penetrated target. Observe that $\mathrm{EU}_A(\sigma_c, \pi_c)$ ranges from 0 to $\alpha_{\max}$, where $\alpha_{\max} = \max_{t \in T} \alpha(t)$. We can also see $\mathrm{EU}_A(\sigma_c, \pi_c)$ as the amount "stolen" by the Attacker. The corresponding *expected Defender's utility for the battery capacity* c defined by

$$\mathrm{EU}_D(\sigma_c, \pi_c) = \alpha_{\max} - \mathrm{EU}_A(\sigma_c, \pi_c)$$

then corresponds to the amount "protected" by σ against π for the battery capacity c.

Note that the Attacker and the Defender have completely antagonistic objectives, i.e., their rationality would remain the same if we defined $\mathrm{EU}_D(\sigma_c, \pi_c)$ as $-\mathrm{EU}_A(\sigma_c, \pi_c)$. However, our present definition of $\mathrm{EU}_D(\sigma_c, \pi_c)$ allows for simple and intuitive interpretation of the corresponding numerical value, and therefore we see this alternative as more convenient.

For reasonable Defender's policies, $\mathrm{EU}_D(\sigma_c, \pi_c)$ increases with increasing c. Hence, the *value of* σ *against* π, defined by

$$Val_G(\sigma, \pi) \;=\; \sup_c \mathrm{EU}_D(\sigma_c, \pi_c)$$

corresponds to the limit protection achieved by σ against π (for a sufficiently large battery capacity). Furthermore, we define the *value of* σ against an arbitrary Attacker's strategy by

$$Val_G(\sigma) = \inf_\pi Val_G(\sigma, \pi),$$

and the *value of* G by $Val_G = \sup_\sigma Val_G(\sigma)$.

Note that Val_G corresponds to the maximal protection achievable for a given patrolling graph G against an Attacker with *unbounded patience* willing to postpone his attack arbitrarily long. Now we give two simple examples demonstrating that considering Attackers with unbounded patience is *unrealistic* and using Val_G as a bound on achievable protection is *misleading*. Intuitively, this is because an optimal attack may correspond to the situation when the remaining energy in the battery drops to some critical level, and the Defender has to prioritize reaching a refill location over the patrolling task. As we shall see, a Defender's policy can be constructed so that the expected waiting time for such a situation is extremely high, and the protection achieved *before* arriving to this situation is substantially higher than Val_G. On the other hand, the Defender cannot postpone visiting a refill location for more than c time units where c is the battery capacity due to the safety restriction. If visiting a refill location requires a specific detour, than executing this detour may provide a good attack opportunity with low expected waiting time. This gives a preliminary intuition behind the dichotomy in attack opportunities formalized in Theorem 2.6.

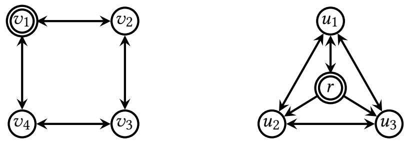


Figure 1: Patrolling graphs of Examples 1 and 2.

Example 1. Consider the patrolling graph of Fig. 1 (left) with four targets v_1, v_2, v_3, v_4, where v_1 is also a refill location. Traversing each edge takes 1 minute and 1 energy unit. All targets are equally important and their importance is equal to 1. The Attacker needs 3 minutes to complete his attack at each target.

If the battery had *infinite* capacity, an *optimal* policy for protecting the targets moves the drone uniformly, i.e., with probability 1/2 to each neighbor target. Thus, *every* attack, initiated at any moment, is discovered with probability at least 1/2, which is the best achievable protection. Here, the Attacker does not gain anything by postponing his attack.

Now suppose the battery has a finite capacity of 120 energy units (two hours flight time), and the battery is fully refilled whenever v_1 is visited. Suppose the drone uses the same policy as above, but whenever the amount of remaining energy units drops to 1, the drone moves to v_1 for a refill (note that thanks to even battery capacity and starting in v_1, the energy level is never 1 in v_3). With probability one, the drone eventually visits v_3 with only 2 units of energy in the battery. Hence, the Attacker can wait until this happens, and attack v_3 as soon as the drone leaves v_3. This attack *succeeds with probability one*, because the drone moves to v_2 or v_4 in the next step, and then *inevitably* to v_1 to refill the battery. So, the achieved protection is *zero* if we assume the Attacker is willing to wait for a good opportunity arbitrarily long (this argument can be generalized to an *arbitrary* policy, i.e., $Val_G = 0$). However, the expected waiting time for such an opportunity is at least 2^{59} minutes, i.e., at least 10^{12} years (the estimated age of the universe is about 13 billion years). Hence, it is fair to say that the considered policy achieves the value 1/2 against a "realistic" Attacker who is not willing to wait so long. □

The next example illustrates that "detours" to refill locations may indeed decrease the achievable protection (even against an "impatient" Attacker who is not willig to postpone his attack for too long) strictly below the level achievable for infinite battery capacity.

Example 2. Consider three targets u_1, u_2, u_3 and one refill location r in the middle (Fig. 1 (right)). The drone needs one minute to move from vertex to vertex, consuming one unit of energy. Completing an attack at u_1, u_2, u_3 takes 3 minutes, and the importance of all targets is equal to 1.

If the battery capacity was infinite, the drone could simply fly around u_1, u_2, u_3 in a circle, discovering every attack in time with probability 1. Now consider a battery with capacity 120 energy units, and let σ be a policy for the drone walking around the circle u_1, u_2, u_3, but whenever u_1 is visited, the drone moves to r with probability 0.3. (Note that u_1 is the only entry point to r.) From r, the drone selects among u_1, u_2, u_3 uniformly at random, and then continues walking around the circle u_1, u_2, u_3 in the way described above. If the amount of remaining energy drops below 4 at u_1, the drone goes to r immediately. If the Attacker's patience is unbounded, this policy achieves *zero* protection, but the expected waiting time for an optimal attack opportunity (the Attacker needs to wait until the drone drains the battery to a low level) is about 3 years. As long as the amount of remaining energy stays above 4 units, the best opportunity available to the Attacker is to wait until the drone decides to move from u_1 to r and attack, e.g., u_1 at this moment. This attack is discovered with probability 2/3, and the Attacker needs to wait about 10 minutes on average for this opportunity. Hence, σ achieves the protection 2/3 against an "impatient" Attacker, which is *optimal* in the sense explained below. □

Now we show how to formalize the protection achievable against an "impatient" Attacker.

Definition 2.5. Let G be a patrolling graph, σ a Defender's policy, and π an Attacker's strategy. Furthermore, for every $c \geq \ell_G$, let $\mathbb{E}^{\sigma_c}[\mathit{Time}^{\pi_c}]$ be the *expected attack time* of π against σ for c. [7]

The expected attack time of π against σ is *polynomial* if there is $n \in \mathbb{N}$ such that $\mathbb{E}^{\sigma_c}[\mathit{Time}^{\pi_c}]$ is $O(c^n)$. The expected attack time of π against σ is *linear* if this holds for $n = 1$.

Clearly, the battery constraint is relevant only if the Attacker is willing to wait longer than the drone's flight time. This means that strategies with *linear* attack time are certainly eligible for an "impatient" Attacker. It is equally clear that strategies with *non-polynomial* (e.g., exponential) expected attack time are *not* eligible for an "impatient" Attacker. However, it is not so clear how to classify Attacker's strategies with expected waiting time bounded by a low degree polynomial (quadratic, cubic, etc.). Should we also see them as eligible? Fortunately, there is no need to resolve this question. The next theorem shows that the maximal level of protection achievable by the Defender against Attacker's strategies with $O(c^n)$ expected attack time, where $n \geq 1$ is a fixed constant, is *independent of n*. In other words, Attacker's strategies with polynomial expected attack time cannot do more harm than strategies with linear expected attack time. The basic intuition behind this result has already been indicated before Example 1—the Defender cannot prevent visiting a refill location in at most c time units, and the associated "detours" provide good opportunities for the (impatient) Attacker with linear expected attack time. There may still exist better attack opportunities, but these are linked to draining the battery below some critical level which can be avoided by the Defender for a very long time (exponential in c). A precise formulation is given in our next theorem.

[7]Our algorithms can easily be adapted also to the setting when the amount of energy consumed along $v \to u$ is *not* proportional to the associated flying time.

Theorem 2.6. *There exists a unique value IVal_G and a parameterized Defender's policy σ^κ satisfying the following:*

A. *For every $\varepsilon > 0$, there is a sufficiently small $\kappa > 0$ such that $\mathit{Val}(\sigma^\kappa, \pi) \geq \mathit{IVal}_G - \varepsilon$ for every Attacker's strategy π with polynomial expected attack time.*
B. *For every $U > \mathit{IVal}_G$ and every Defender's policy σ, there exists an Attacker's strategy π with linear expected attack time such that $\mathit{Val}(\sigma, \pi) \leq U$.*

Part A. of Theorem 2.6 says that the Defender can achieve protection arbitrarily close to *IVal* against an Attacker who is not willing to wait exponentially long (technically, we take into account only strategies with polynomial expected attack time). According to B., the Defender cannot achieve a protection P *strictly better* than *IVal*, because for $U = (P + \mathit{IVal})/2$, the Attacker has a strategy with linear expected attack time such that the achieved protection is at most U, i.e., strictly smaller than P.

The policy σ^κ is "parameterized" by κ in the sense that certain probability distributions used by σ^κ depend on the concrete value of κ (as we shall see, κ corresponds to the probability of scheduling a "detour" to a refill location). Intuitively, the value *IVal* and the policy σ^κ are obtained by first considering optimal moving strategies (*without* battery constraints) in certain extensions of G, and selecting the graph where the associated optimal strategy γ achieves the best protection. *IVal* is then defined as the value of γ, and the policy σ^κ is obtained by modifying γ. For the sake of clarity, in the following proof we disregard the case when the importance of some targets is so low that an optimal moving strategy can avoid visiting them completely (we indicate the place where this assumption is needed, and sketch the required technical adjustment).

A proof of Theorem 2.6. For every edge $v \to r$ of G, where $v \in T$ and $r \in R$, we define a game graph $G_{v \to r}$ obtained from G by adding a fresh non-target vertex $\hat{v}$ and an edge $\hat{v} \to r$ such that $t(\hat{v} \to r) = t(v \to r)$.

Let γ be a moving strategy in $G_{v \to r}$ initiated in $\hat{v}$, i.e., $\gamma(\varepsilon)$ selects $\hat{v}$ with probability 1. For every history $h = v_1, \ldots, v_n$, let $\gamma{\downarrow}h$ be a moving strategy initiated in v_n such that for every history $u_1, \ldots, u_m$ initiated in v_n we have that $\gamma{\downarrow}h(u_1, \ldots, u_m) = \gamma(v_1, \ldots, v_n, u_2, \ldots, u_m)$. That is, $\gamma{\downarrow}h$ behaves like γ after performing h. Furthermore, for every $z \in T$, let ϱ_z be a function assigning attack_z to every "one-step" observation of the form $t, t \to u$. That is, ϱ_z says "attack z immediately".

A λ-*opportunity* against γ (in $G_{v \to r}$) is a history $h = v_1, \ldots, v_n$ initiated in $\hat{v}$ such that $\mathcal{P}^\gamma(h) > 0$ and there exists $z \in T$ satisfying $\mathcal{P}^{\gamma{\downarrow}h, \varrho_z}(\mathit{penetrate}_z) \cdot \alpha(z) \geq \lambda$. Let $\Lambda_\gamma[v \to r]$ be the supremum of all λ's such that there exists a λ-opportunity against γ. We define

$$\Lambda[v \to r] \ := \ \inf\{\Lambda_\gamma[v \to r] \mid \gamma \text{ is initiated in } \hat{v}\}\ .$$

From now on, we assume that the edge $v \to r$ has been chosen so that $\Lambda[v \to r]$ is *minimal.* We put $\mathit{IVal}_G := \alpha_{\max} - \Lambda[v \to r]$. Furthermore, let γ be an *optimal* moving strategy for $G_{v \to r}$ satisfying $\Lambda_\gamma[v \to r] = \Lambda[v \to r]$. The policy σ^κ is obtained from γ in two steps.

(1) We modify γ into γ^κ behaving identically as γ but whenever v is visited[8], γ^κ selects the edge $v \to r$ with probability κ

[8]Here we use our assumption that no target (incl. v) can be "systematically bypassed" by γ because of its low importance. Without this assumption, we need to use a combination of optimal moving strategies for several $G_{v \to r}$ instead of γ, and then

(the probabilities of selecting the other edges are adjusted). When the edge $v \rightarrow r$ is performed, γ^{κ} restarts simulating γ. This construction ensures that for every $\varepsilon > 0$, there is $\kappa > 0$ such that $\Lambda_{\gamma^{\kappa}}[v{\rightarrow}r] \leq \Lambda[v{\rightarrow}r] + \varepsilon/2$.

(2) The moving strategy γ^{κ} is "folded" into another moving strategy $\hat{\gamma}^{\kappa}$ as follows. First, $\hat{\gamma}^{\kappa}$ executes γ^{κ} until some vertex u is revisited and the behavior of γ^{κ} in the two occurrences of u is "ε/Γ-similar" in the next $d_{\max}$ steps, where Γ is a suitable constant depending only on G. Technically, this means that the probabilities of visiting every vertex in precisely k steps (where $k \leq d_{\max}$) from the two occurrences of u differ at most by ε/Γ. By the pigeonhole principle, such a pair of vertices must occur along every infinite path. After encountering such a pair, the strategy $\hat{\gamma}^{\kappa}$ "folds back" and restarts to simulate γ^{k} from the *first* occurrence of u. One can show that by choosing a sufficiently large Γ, we obtain $\Lambda_{\hat{\gamma}^{\kappa}}[v{\rightarrow}r] \leq \Lambda[v{\rightarrow}r] + \varepsilon$. Moreover, $\hat{\gamma}^{\kappa}$ is a *regular*[9] strategy [17].

For every vertex v, let $\iota(v)$ be the *critical energy level* defined as the least ℓ such that for every edge $v{\rightarrow}u$ there exists a path from u to r of length at most $\ell - t(v{\rightarrow}u)$. The policy σ^{κ} is obtained from the moving strategy $\hat{\gamma}^{\kappa}$ as follows: For every c, the strategy σ^{κ}_{c} behaves like $\hat{\gamma}^{\kappa}$ until the battery is drained to a critical level where σ^{κ}_{c} strives to reach r along the shortest path. After refilling the battery, the simulation of $\hat{\gamma}^{\kappa}$ is restarted.

Since $\Lambda_{\hat{\gamma}^{\kappa}}[v{\rightarrow}r] \leq \Lambda[v{\rightarrow}r] + \varepsilon$ and σ^{κ} behaves identically as $\hat{\gamma}^{\kappa}[v{\rightarrow}r]$ until reaching a critical energy level, every Attacker's strategy π such that $Val(\sigma^{\kappa}, \pi) < IVal_G - \varepsilon$ must "wait" until the remaining energy drops to a critical level with probability bounded away from zero (for every battery capacity). Since the expected time for reaching a critical battery level is clearly exponential in c, the expected attack time of π is also exponential in c, which proves Theorem 2.6.A.

Theorem 2.6.B is obtained by showing the expected waiting time for an optimal attack opportunity against $\hat{\gamma}^{\kappa}[v{\rightarrow}r]$ is linear in c. Here we use the regularity of $\hat{\gamma}^{\kappa}[v{\rightarrow}r]$. □

Example 3. For the patrolling graphs of Examples 1 and 2, we have that $Val_G = 0$. If we further assume that the importance of all targets is 1, then $IVal_G$ is equal to $1/2$ and $2/3$, respectively. Also observe that if the battery capacity is increased from 120 units to 180 units in Example 2 (i.e., 3 hours flight time instead of 2 hours), then the lower bound on the expected waiting time for an attack opportunity better than $2/3$ increases from 3 years to 3750 years (this demonstrates the exponential growth of the expected waiting time for an attack opportunity better than *IVal*). □

The next example shows that *IVal* is not necessarily achievable, even if the battery capacity is arbitrarily large.

Example 4. Consider a graph of Fig. 2 where $d(v_i) = 3$ and $\alpha(v_i) = 1$ for all $i \in \{1, 2, 3\}$. We have that $IVal_G = 1/2$, because for every $\varepsilon > 0$, we can construct a policy σ^{ε} such that σ^{ε}_{c} selects the edge $v_2 \rightarrow r$ with probability 2ε and the edges $v_2 \rightarrow v_1$ or $v_2 \rightarrow v_3$ with probability $1/2 - \varepsilon$ (for every c). Note σ^{ε} achieves protection $1/2 - \varepsilon$.

modify this strategy by including "detours" to refill locations. The idea is the same as for γ and γ^{κ}, but the construction is more technical.

[9]A *regular* strategy uses finitely many memory elements to collect information about the history, and the decision taken by the strategy depends only on this finite information. We refer to [17] for details.

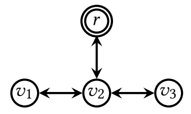


Figure 2: A patrolling graph G.

Now suppose there exist σ and c such that for every Attacker's strategy π with polynomial expected attack time we have that $\mathrm{EU}_D(\sigma_c, \pi_c) = 1/2$. During the first c moves, the drone must inevitably visit v_1 or v_3 in situation when σ_c selects the edge $v_2 \rightarrow r$ with some positive probability *in the next step*. This means that either $v_2 \rightarrow v_1$ or $v_2 \rightarrow v_3$ is selected with probability strictly smaller than $1/2$ in the next step. The Attacker can recognize this opportunity (he knows σ_c and observes the drone). So, he can perform an attack succeeding with probability strictly larger than $1/2$ (during the first c moves of the drone), which is a contradiction. □

Finally, note that deciding whether $IVal = 1$ is **NP**-hard due to a trivial reduction of the Hamiltonian cycle problem. A given graph with n vertices can be seen as a patrolling graph with n targets where all targets are also refill locations and $d(v) = n$, $\alpha(v) = 1$ for every vertex v. Traversing every edge takes 1 unit of time. Then, $IVal_G = 1$ iff there exists a Hamiltonian cycle in the graph. Furthermore, if $IVal \neq 1$, then $IVal \leq 1 - 1/n$, and hence it is **NP**-hard to distinguish whether $IVal = 1$ or $IVal \leq 1 - 1/n$.

3 EFFICIENT POLICY SYNTHESIS

In this section, we design a polynomial-time algorithm computing a regular[10] Defender's policy against an impatient Attacker for a given patrolling graph $G = (V, T, R, \rightarrow, t, d, \alpha)$. The algorithm basically "mimics" the construction of *IVal* and σ^{κ} described in the proof of Theorem 2.6. We start by introducing two useful notions.

- For every Defender's policy σ, let $IVal(\sigma) = \inf_{\pi} Val(\sigma, \pi)$ where π ranges over all Attacker's strategies with polynomial expected attack time. Hence, $IVal(\sigma)$ is the protection achieved by σ against an impatient Attacker.
- For every $\tau \in \mathbb{N}$, let $c[\sigma, \tau]$ be the least capacity such that the expected time accumulated along a λ-opportunity against $\sigma_{c[\sigma,\tau]}$, where $\lambda > \alpha_{\max} - IVal(\sigma)$, is at least τ.

The algorithm inputs

$$G = (V, T, R, \rightarrow, t, d, \alpha), \quad \varepsilon > 0, \quad \tau \in \mathbb{N}$$

and outputs

$$V_{\min}, \quad V_{\max}, \quad (\sigma, IVal(\sigma)), \quad c_{\max}$$

where $V_{\min}$ and $V_{\max}$ are lower and upper bounds on $IVal_G$, σ is a policy such that $IVal(\sigma) \geq V_{\min} - \varepsilon$, and $c_{\max}$ is an upper bound on $c[\sigma, \tau]$.

Computing $V_{\min}$. For every $v \rightarrow r$, our algorithm starts by computing a moving strategy γ for $G_{v\rightarrow r}$ initiated in $\hat{v}$ such that $\Lambda_{\gamma}[v{\rightarrow}r]$ is as low as possible. Here we use a slightly adjusted version of the polynomial-time algorithm presented in [17]. Then,

[10]A moving strategy may generally depend on the whole history h of Defender's moves. A *regular* moving strategy uses finitely many memory elements to collect some information about h and the decision depends just on the current memory state. We refer to [17] for details.

the algorithm identifies the edge $v \to r$ and the associated moving strategy γ such that $\Lambda_\gamma[v{\to}r]$ is *minimal* and sets

$$V_{\min} \ := \ \alpha_{\max} - \Lambda_\gamma[v{\to}r]$$

Computing the policy σ. The policy σ is obtained by modifying γ in the following way. First, the algorithm checks whether γ *always* selects the edge $v \to r$ with some positive probability whenever visiting v. In this case, there is *no need* to perform the modification of γ into γ^κ described in item **(1)** in the proof of Theorem 2.6. Otherwise, the algorithm computes a sufficiently small κ such that $\mathit{IVal}(\gamma^\kappa)$ stays ε-close to $V_{\min}$ and performs the modification (note that $\mathit{IVal}(\gamma^\kappa)$ is a monotone function of κ, so we can use binary search to find an admissible value of κ). Since γ is regular, the "folding" described in item **(2)** in the proof of Theorem 2.6 is unnecessary (whenever a given u is visited, there are only finitely many possibilities how γ may behave). Then, the policy σ is obtained from γ (or γ^κ) by changing its behaviour when draining the battery to a critical level (see the proof of Theorem 2.6).

Computing $V_{\max}$. Using the results of [17], the algorithm computes an upper bound on $\alpha_{\max} - \Lambda[v{\to}r]$ (for every $v{\to}r$) and sets $V_{\max}$ to the maximum of these upper bounds.

Computing $c_{\max}$. We design an algorithm computing, for a given battery capacity $c \in \mathbb{N}$, a lower bound b_c on the expected waiting time for a λ-opportunity where $\lambda > \alpha_{\max} - \mathit{IVal}(\sigma)$. Since b_c is a monotone function of c, we can use binary search to find the least $\hat{c}$ such that $b_{\hat{c}} \geq \tau$. Since $\hat{c} \geq c[\sigma, \tau]$, we set $c_{\max} := \hat{c}$.

The way of computing the bound b_c is technical, and here we present just the main underlying idea. For simplicity, assume that the policy σ computed above uses just one memory element (i.e., σ is positional), there is only one refill location r, and traversing every edge in the game graph takes one time unit[11]. Let ι be a (minimal) battery capacity such that the drone can still safely arrive to a refill location after performing one edge in an arbitrary vertex (i.e., σ is not forced to go for a refill as long as the battery level is kept above ι). Let X be a random variable assigning to every walk the return time to r. Furthermore, let $c' = c - \iota$ and $p = \mathbb{P}[X > c']$ denote the probability that the battery gets drained to the critical level before refilling in r. Then, the expected waiting time for a λ-opportunity where $\lambda > \alpha_{\max} - \mathit{IVal}(\sigma)$ is at least $\mathbb{E}[X \mid X \leq c'] \cdot 1/p$ (the factors correspond to the expected time between consecutive refillings and the expected number of refillings before the battery gets drained, respectively). Let P denote the transition matrix of σ with an additional absorbing vertex r' and each transition to r redirected to r'. If c' is small enough, we compute $P, P^2, P^3, \ldots, P^{c'}$. Then, we have $p = 1 - (P^{c'})_{rr'}$ and also

$$\mathbb{E}[X \mid X \leq c'] \ = \ \frac{1}{\mathbb{P}[X \leq c']} \cdot \sum_{i=1}^{c'} i \cdot \mathbb{P}[X = i]$$

where $\mathbb{P}[X \leq c'] = 1 - p$ and $\mathbb{P}[X = i] = (P^i)_{rr'} - (P^{i-1})_{rr'}$. The running time of this procedure is linear in c'; if this is infeasible, we instead compute a 2-approximation of the above as follows: Let $k \in \mathbb{N}$ be the largest such that $2^k \leq c'$. By repeated squaring, we compute $P, P^2, P^4, P^8, \ldots, P^{2^k}$ and using the binary representation of c', we multiply the corresponding matrices to get $P^{c'}$. Then, we have that $\sum_{i=1}^{c'} i \cdot \mathbb{P}[X = i]$ is at least

$$\mathbb{P}[X{=}1] + \left(\sum_{j=0}^{k-1} 2^j \cdot \mathbb{P}[2^j < X \leq 2^{j+1}] \right) + 2^k \cdot \mathbb{P}[2^k < X \leq c']$$

where $\mathbb{P}[X = 1] = P_{rr'}$, $\mathbb{P}[2^j < X \leq 2^{j+1}] = (P^{2^{j+1}})_{rr'} - (P^{2^j})_{rr'}$, and similarly for the last probability.

Notes. Our policy synthesis algorithm can apply an *arbitrary* procedure for solving adversarial patrolling (without battery constraints) to $G_{v\to r}$, as long as this procedure produces a regular moving strategy. We employed the procedure of [17] because it is guaranteed to terminate quickly and produces efficient strategies. Nevertheless, our approach to solving adversarial patrolling with drones can immediately utilize any progress in solving adversarial patrolling *without* battery constraints.

In [26], Attacker's impatience is modeled by discounting the Attacker's utility. According to Theorem 2.6, for a sufficiently large c there is a suitable discount factor δ such that the Stackelberg value in the discounted model is close to *IVal*. However, it is not clear how to compute such a δ without running our algorithm.

4 EXPERIMENTS

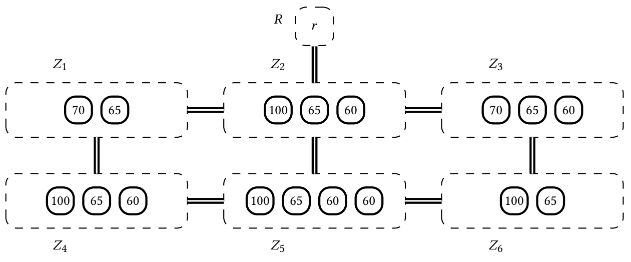
In this section, we present the outcomes of our algorithms for nontrivial instances of the patrolling problem. All experiments reported in this section were run on the same machine equipped with an Intel Core i7-4810MQ CPU and 16 GB of RAM.

4.1 Experiment 1

We consider a patrolling graph whose structure is shown in Fig. 3. There are 17 targets in 6 zones $Z_1, \ldots, Z_6$. Local flights are forbidden in each zone (i.e., a drone cannot fly between targets within the same zone). All flights among targets in different zones must stay within designated corridors connecting these zones indicated by double lines. For example, it is possible to fly freely between the targets of Z_5 and Z_6 in both directions, but there is no corridor for flying directly from Z_1 to Z_5. A flight along every corridor takes one minute. The corridors are narrow so we assume the Attacker can only determine the next zone visited by the drone (but not a concrete target in this zone) when the drone leaves the currently visited zone. The corresponding patrolling graph models this feature by adding auxiliary vertices and edges. Each target has importance either 100, 70, 65, or 60. For example, Z_3 contains three targets of importance 75, 65, and 60. Completing an attack at each target takes 6.5 minutes. There is only one refill location r in a separate zone R.

We aim at solving *four* patrolling problems P_{100}, P_{70}, P_{65}, and P_{60}, where the task of P_i is to protect precisely the targets of importance at least i (hence, the P_i's model different priority levels). All zones containing only the targets whose importance is strictly smaller than i become *forbidden* zones in P_i, i.e., the drone is not allowed to fly through them. In particular, in P_{60} we aim at protecting *all* targets, and in P_{100} we aim to protect only the four targets of importance 100 contained in Z_2, Z_4, Z_5, and Z_6. The table of Fig. 3 shows the values of the regular policies $\sigma[j]$ computed by our

[11]If these assumptions are not satisfied, we perform preliminary adjustments. In particular, a finite-memory policy is fully specified by its behaviour in pairs of the form (v, m), where v is a vertex and m a memory element, so it can be encoded by a "transition matrix" where the rows/columns are indexed by these pairs.



	P_{100}	P_{70}	P_{65}	P_{60}
$IVal(\sigma[1])$	55.5	58.8	44.9	43.2
$IVal(\sigma[2])$	61.7	65	49.2	45.6
$IVal(\sigma[3])$	99	65	49.2	45.6
$V_{\max}$	100	74.3	53.2	48.5

Figure 3: Patrolling 17 targets in 6 zones.

algorithm, where j indicates the number of internal memory elements of the regular strategy $\sigma[j]$. In particular, $\sigma[1]$ is a *positional* strategy obliged to perform the same decision when revisiting the same vertex. The row $V_{\max}$ shows the upper bound on *IVal*.

Note that for P_{100}, three memory elements are sufficient to achieve an *almost optimal* strategy ($IVal(\sigma[3])$ is very close to $V_{\max}$). Intuitively, this is because an efficient strategy inevitably needs to perform a different decision in Z_5 depending on which zone the drone came from (there are three possibilities). Increasing j above 3 does not lead to any improvement in the achieved protection.

In all cases, we set τ to ten years and $\varepsilon = 1$. The obtained bounds on $c[\sigma, \tau]$ indicate that seven-hour flight time is sufficient to push the expected waiting time for a λ-opportunity where $\lambda > \alpha_{\max} - IVal(\sigma[i])$ beyond 10 years (for all i's).

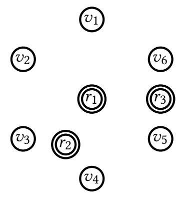
4.2 Experiment 2

In our second experiment, we consider a *parameterized* patrolling graph and evaluate our algorithm for increasing values of the parameter. For every $n \geq 4$, let G_n be a patrolling graph with n vertices, where $n{-}1$ vertices are targets arranged in a circle so that the distance between two neighboring targets is the same and equal to 1. Furthermore, there is one refill location in the middle. The graph is fully connected, and the distance between each pair of vertices is the Euclid distance. For every target v, the importance $\alpha(v)$ is chosen randomly in the interval $\langle 500, 1000\rangle$, and $d(v)$ is chosen randomly in the interval $\langle n, 2\varrho{+}n{-}1)$ where ϱ is the radius of the circle (the choice of $d(v)$ prevents the existence of "trivial" optimal policies). The drone can move along an edge of length x in x minutes, and one energy unit corresponds to 1 minute flight time.

The outcomes of our algorithm for G_n, where $n \in \{4, \ldots, 30\}$, $\varepsilon = 1$, and $\tau = 5\,256\,000$ (10 years) are shown in Table 1. In all cases, $IVal(\sigma) = V_{\min}$ (cf. the previous section). We also report the protection Val^∞ achieved by a policy computed for *infinite* battery capacity by the algorithm of [17]. The last column gives an upper bound on $\sigma_{c[\sigma,\tau]}$ translated into the required flight time in hours (recall that one unit of energy corresponds to one minute flight time). That is, the last column is an upper bound on the flight time needed to push the expected waiting time for a λ-opportunity where $\lambda > \alpha_{\max} - IVal(\sigma)$ beyond 10 years. For $n = 30$, this flight time is almost 4 days, which is still achievable by advanced UAVs used in security applications. Since the procedure for computing $V_{\max}$ is exponential, we succeeded to produce the value only for $n \leq 12$ (limiting computation time to 2 hours).

n	$IVal(\sigma) = V_{\min}$	$V_{\max}$	Val^∞	*Flight time (hrs)*
4	617.313	618.849	826.274	1.250
5	603.670	606.143	765.675	1.967
6	838.174	842.061	985.541	3.483
7	813.657	818.493	943.325	4.233
8	886.528	890.754	981.047	7.067
10	887.344	890.029	962.008	11.300
12	926.070	928.321	989.464	15.817
15	921.639	–	973.265	24.083
20	954.532	–	990.765	43.233
25	961.775	–	993.186	61.117
30	938.818	–	962.954	88.767

Table 1: Experimental results for G_n, ε=1, and τ=10 years.



$d(v_i)$	r_1	r_2	r_3
3	$\frac{1}{6}$	0	0
6	$\frac{5}{6}$	$\frac{5}{6}$	1
6.5	$\frac{5}{6}$	1	1

Figure 4: Computing an optimal position for a refill location.

4.3 Experiment 3

Here we show that our algorithm can also be used to identify an optimal position for a refill location. Consider the patrolling graph of Fig. 4. There are 6 targets $v_1, \ldots, v_6$ arranged regularly around a circle with radius 1. The graph is fully connected, and the distance is the Euclid distance. Furthermore, there are three eligible positions for a refill location r_1, r_2, r_3, where the distance of r_2 from the center is 2/3. The table of Fig. 4 shows $IVal(\sigma)$ for the regular policy σ computed by our algorithm when the *only* refill location is either r_1, r_2, or r_3, and the attack length at every v_i is set either to 3, 6, or 6.5. Note that *none* of the considered refill locations is optimal for *all* attack lengths.

5 FUTURE WORK

A natural continuation of our work is considering scenarios with *multiple* Defenders/Attackers. A straightforward idea is to assign a subset of targets to each Defender and compute a tuple of policies using our algorithm. However, security applications may impose additional restrictions on robustness, resilience, etc., making the problem challenging.

ACKNOWLEDGEMENT

The authors are supported by the Czech Science Foundation, grant No. 18-11193S. The work is also supported by GAMENET, the European Network for Game Theory (COST Action 16228).

REFERENCES

[1] N. Agmon, S. Kraus, and G. Kaminka. 2008. Multi-Robot Perimeter Patrol in Adversarial Settings. In *Proceedings of ICRA 2008*. 2339–2345.

[2] N. Agmon, V. Sadov, G. A. Kaminka, and S. Kraus. 2008. The impact of adversarial knowledge on adversarial planning in perimeter patrol. In *Proceedings of AAMAS 2008*. 55–62.

[3] N. Basilico, N. Gatti, and F. Amigoni. 2009. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *Proceedings of AAMAS 2009*. 57–64.

[4] N. Basilico, N. Gatti, and F. Amigoni. 2012. Patrolling Security Games: Definitions and Algorithms for Solving Large Instances with Single Patroller and Single Intruder. *Artificial Inteligence* 184–185 (2012), 78–123.

[5] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. 2009. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *Proceedings of WI-IAT 2009*. 557–564.

[6] N. Basilico, N. Gatti, and F. Villa. 2010. Asynchronous Multi-Robot Patrolling against Intrusion in Arbitrary Topologies. In *Proceedings of AAAI 2010*. 1224–1229.

[7] N. Basilico, G. De Nittis, and N. Gatti. 2016. A Security Game Combining Patrolling and Alarm-Triggered Responses Under Spatial and Detection Uncertainties. In *Proceedings of AAAI 2016*. 404–410.

[8] B. Bošanský, V. Lisý, M. Jakob, and M. Pěchouček. 2011. Computing Time-Dependent Policies for Patrolling Games with Mobile Targets. In *Proceedings of AAMAS 2011*. 989–996.

[9] B. Bošanský, O. Vaněk, and M. Pěchouček. 2012. Strategy Representation Analysis for Patrolling Games. In *Proceedings of AAAI Spring Symposium 2012*. 9–12.

[10] E. Munoz de Cote, R. Stranders, N. Basilico, N. Gatti, and N. Jennings. 2013. Introducing alarms in adversarial patrolling games: extended abstract. In *Proceedings of AAMAS 2013*. 1275–1276.

[11] F. Fang, A. X. Jiang, and M. Tambe. 2013. Optimal Patrol Strategy for Protecting Moving Targets with Multiple Mobile Resources. In *Proceedings of AAMAS 2013*. 957–964.

[12] J. Gan, B. An, Y. Vorobeychik, and B. Gauch. 2017. Security Games on a Plane. In *Proceedings of AAAI 2017*. AAAI Press, 530–536.

[13] E. Hartuv, N. Agmon, and S. Kraus. 2018. Scheduling Spare Drones for Persistent Task Performance under Energy Constraints. In *Proceedings of AAMAS 2018*. 532–540.

[14] M. Jain, E. Karde, C. Kiekintveld, F. Ordóñez, and M. Tambe. 2010. Optimal Defender Allocation for Massive Security Games: A Branch and Price Approach. In *Proceedings of OptMAS 2010*. 1–8.

[15] J. Karwowski, J. Mandziuk, A. Zychowski, F. Grajek, and B. An. 2019. A Memetic Approach for Sequential Security Games on a Plane with Moving Targets. In *Proceedings of AAAI 2019*. 970–977.

[16] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, F. Ordóñez, and M. Tambe. 2009. Computing Optimal Randomized Resource Allocations for Massive Security Games. In *Proceedings of AAMAS 2009*. 689–696.

[17] D. Klaška, A. Kučera, T. Lamser, and V. Řehák. 2018. Automatic Synthesis of Efficient Regular Strategies in Adversarial Patrolling Games. In *Proceedings of AAMAS 2018*. 659–666.

[18] E. S. Lin, N. Agmon, and S. Kraus. 2019. Multi-robot adversarial patrolling: Handling sequential attacks. *Artificial Intelligence* 274 (2019), 1 – 25.

[19] S. Mishra, S. Rodríguez, M. Morales, and N. M. Amato. 2016. Battery-constrained coverage. In *Proceedings of CASE 2016*. 695–700.

[20] T. Nguyen and T. Au. 2017. Extending the Range of Delivery Drones by Exploratory Learning of Energy Models. In *Proceedings of AAMAS 2017*. 1658–1660.

[21] J. Pita, M. Jain, J. Marecki, F. Ordónez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. 2008. Deployed ARMOR Protection: The Application of a Game Theoretic Model for Security at the Los Angeles Int. Airport. In *Proceedings of AAMAS 2008*. 125–132.

[22] A. Rosenfeld, O. Maksimov, and S. Kraus. 2018. Optimal Cruiser-Drone Traffic Enforcement Under Energy Limitation. In *Proceedings of IJCAI 2018*. 3848–3855.

[23] G. P. Strimel and M. M. Veloso. 2014. Coverage planning with finite resources. In *In Proc. of 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2950–2956.

[24] A. Sugiyama, L. Wu, and T. Sugawara. 2018. Learning of Activity Cycle Length based on Battery Limitation in Multi-agent Continuous Cooperative Patrol Problems. In *Proceedings of ICAART 2018,*. INSTICC, 62–71.

[25] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. 2009. IRIS—A Tool for Strategic Security Allocation in Transportation Networks Categories and Subject Descriptors. In *Proceedings of AAMAS 2009*. 37–44.

[26] Y. Vorobeychik, B. An, and M. Tambe. 2012. Adversarial Patrolling Games. In *Proceedings of AAAI Spring Symposium 2012*. 91–98.

[27] H. Xu, A. X. Jiang, A. Sinha, Z. Rabinovich, S. Dughmi, and M. Tambe. 2015. Security Games with Information Leakage: Modeling and Computation. In *Proceedings of IJCAI 2015*. 674–680.

[28] H. Xu, K. Wang, P. Vayanos, and M. Tambe. 2018. Strategic Coordination of Human Patrollers and Mobile Sensors With Signaling for Security Games. In *Proceedings of AAAI 2018*. 1290–1297.