Optimal Patrol to Uncover Threats in Time When Detection Is Imperfect

Kyle Y. Lin[*], Michael Atkinson[†], Kevin D. Glazebrook[‡]

May 12, 2014

Abstract

Consider a patrol problem, where a patroller traverses a graph through edges to detect potential attacks at nodes. An attack takes a random amount of time to complete. The patroller takes one time unit to move to and inspect an adjacent node, and will detect an ongoing attack with some probability. If an attack completes before it is detected, a cost is incurred. The attack time distribution, the cost due to a successful attack, and the detection probability all depend on the attack node. The patroller seeks a patrol policy that minimizes the expected cost incurred when, and if, an attack eventually happens. We consider two cases. A random attacker chooses where to attack according to predetermined probabilities, while a strategic attacker chooses where to attack to incur the maximal expected cost. In each case, computing the optimal solution, although possible, quickly becomes intractable for problems of practical sizes. Our main contribution is to develop efficient index policies—based on Lagrangian relaxation methodology, and also on approximate dynamic programming—which typically achieve within 1% of optimality with computation time orders of magnitude less than what is required to compute the optimal policy for problems of practical sizes.

Keywords: surveillance, infrastructure protection, search and detection, Lagrangian realxation, approximate dynamic programming.

1 Introduction

While patrol problems have been studied since the 1950s [12], there has been renewed interest in this subject due to the rapid advancement of surveillance technology in recent years, such as unmanned vehicles, automatic image recognition, and data fusion. Most of the earlier works on patrol planning assume that patrol forces are allocated to maximize some

[*]Operations Research Department, Naval Postgraduate School, Monterey, CA 93943, kylin@nps.edu

[†]Operations Research Department, Naval Postgraduate School, Monterey, CA 93943, mpatkins@nps.edu

[‡]Department of Management Science, Lancaster University Management School, Lancaster University, Lancaster LA1 4YX, United Kingdom, k.glazebrook@lancaster.ac.uk

performance measure based on known and fixed frequencies of illicit activities at different locations [7, 8, 9, 13, 14, 16, 23]. In other words, these works do not account for the possibility that the adversary may change his behavior in the presence of patrol.

There has been a growing interest in recent years in taking a game-theoretic approach to modeling patrol problems. With an intelligent attacker who seeks to minimize his probability of getting caught, the objective of the patrol force is to determine a patrol policy—possibly randomized—to maximize the minimum detection probability regardless of where the attacker chooses to attack. A common model framework is to embed the patrol area in a graph, with nodes representing potential targets for attack, and edges connecting targets next to each other. For instance, a museum can be divided into exhibit rooms as nodes and connecting doors as edges; an open area can be divided into hexagon cells as nodes and adjacent cells connected by edges. A patroller then has to decide how to traverse the graph through edges to detect potential attacks at nodes [3, 4, 5, 15, 22]. There is also a stream of works that study patrol problems with multiple agents. These works typically use distributed methods that rely on local objective functions and myopic policies to produce scalable patrol strategies [1, 2, 6, 17, 18, 19, 24].

In this paper, we use a graph to model the patrol area, with one patroller traversing the edges in the graph in order to detect potential attacks at nodes, similar to the framework in Alpern et al. [3] and Lin et al. [15]. There are two prominent features of our model: (1) the time it takes to attack a node is a random variable whose probability distribution depends on the node, and (2) the patroller may overlook an ongoing attack at the inspected node. Whereas most of the earlier works formulate a mathematical program to determine the optimal patrol strategy, these two features make the optimal solution computationally feasible only for small graphs. To the best of our knowledge, our work is the first to address these two features simultaneously.

Specifically, our work extends that of Lin et al. [15] by allowing the possibility of overlooking. In other words, when the patroller inspects a node, he will detect an ongoing attack with a probability that depends on the node. For instance, an unmanned aerial vehicle may have a better chance to locate a target in desert than in forest. Mathematically, the possibility of overlooking makes the problem considerably more difficult. To decide where to go next, it is no longer sufficient for the patroller to only keep track of the last time he inspected each of the nodes. Thus the methods reported in Lin et al. [15] no longer apply.

We first analyze the case of *random attackers*, who choose which node to attack according to a probability distribution. Although it is possible to formulate the problem as a Markov decision process and to compute the optimal solution via a linear program, computing the optimal policy quickly becomes intractable for problems of practical sizes. The main contribution of this paper is to introduce two methods to develop heuristic policies that score nodes as candidates to be inspected next by giving them each an index.

1. The first method uses Lagrangian relaxation to develop a node index which can be interpreted as the fair charge for a patrol inspection at that node in its current state. Because in general an arbitrary state cannot map directly to such a charge, the novelty of our method is to use both the *expected number of ongoing attacks* and *their departure rates in the near future* as a conduit to develop an index.

2. The second approach to index generation uses approximate dynamic programming. Specifically, we first compute a lower bound for the optimal policy and use the lower bound to infer the patrol rate at each node. The novelty of this method is to approximate the value of inspecting a node by assuming the future patrols will arrive at rates implied from the lower bound.

In our numerical experiments, these index policies typically achieve within 1% of optimality with computation time orders of magnitude less than what is required to compute the optimal policy for problems of practical sizes. These index policies also allow us to construct effective patrol strategies against *strategic attackers*, who seek to maximize the expected damage by attacking the most vulnerable node.

The rest of this paper proceeds as follows. Section 2 introduces a patrol model and a linear program to compute the optimal solution. Section 3 presents index policies based on Lagrangian relaxation, and Section 4 presents index policies based on approximate dynamic programming. Section 5 presents numerical results for these index heuristics, and Section 6 demonstrates how they can be used to construct effective patrol strategies against strategic attackers. Finally, Section 7 concludes the paper.

2 The Model

This section extends the graph patrol model studied in [15], so that a patroller may overlook an attack when they are at the same location. The patrol area is divided into n locations that are subject to enemy attacks, with each location represented by a node and adjacent locations connected by an edge. A patroller traverses the edges in the graph trying to detect attacks at nodes. It takes 1 time unit for the patroller to inspect a node, and at the end of the inspection, the patroller can move to an adjacent node (or stay at the same node) and inspect it. In other words, a patrol schedule is a sequence of nodes that observe the edge constraints in the graph.

We first consider the case of random attackers, who will attack node i with probability p_i upon arrival. The time it takes to complete an attack at node i is random and follows cumulative distribution function $F_i(\cdot)$, for $i = 1, \ldots, n$. If the patroller inspects node i, then at the end of the inspection the patroller will detect an ongoing attack with probability α_i, or overlook it with probability $1 - \alpha_i$, independent of everything else, for $i = 1, \ldots, n$. A cost c_i is incurred if an attack completes at node i before being detected; otherwise no cost is incurred.

We assume that the patroller has no knowledge about when an attack will occur, so a sensible objective is to minimize the expected cost incurred when, and if, an attack eventually happens. Mathematically, we seek to determine the patrol policy that minimizes the expected cost when an attack occurs in the system's steady state. To do so, we assume the attackers arrive according to a Poisson process with rate Λ, with each attacker operating independently, and possibly simultaneously at the same node. By letting the patrol process continue indefinitely without interruption by attacks—whether an attack is detected or not—we seek the patrol policy that minimizes the long-run cost rate. Because this long-run cost rate scales proportionally in Λ, the optimal policy does not depend on Λ. In fact, the patrol policy that minimizes the *long-run cost rate* also minimizes the *long-run average cost for each attack*, therefore the *expected cost due to an attack in steady state.*

2.1 Markov Decision Process Formulation

To make it possible to formulate the problem as a Markov decision process, we assume that the attack time distribution $F_i(\cdot)$ is bounded by B_i, for $i = 1, \ldots, n$, and let $B = \lceil \max_i B_i \rceil$. To formulate the problem, we define time 0 as the time of the next detection opportunity. Because all attackers that arrived at time $-B$ or earlier would have completed their attacks by time 0, if not detected, the patroller only needs to keep track of what happened in the time interval $[-B, 0)$ in order to decide what to do at time 0. Presumably, the information gathered in the interval $[-B, 0)$ includes the nodes inspected and the inspection results. It turns out, however, that knowing where the patroller has been in the interval $[-B, 0)$ is sufficient; the additional information about the inspection results does not help the patroller make a better decision at time 0. This result follows from the fact that the attackers arrive according to a Poisson process and that each attacker acts independently, as stated in the next theorem.

Theorem 1 The optimal patrol policy—which node to inspect at time 0—depends only on where the patroller has been in $[-B, 0)$, but not on the number of attacks detected in each of those patrol inspections.

Proof. Suppose that the patroller just completed an inspection at time -1 and needs to decide which adjacent node to inspect at time 0. For any given patrol history—namely which node the patroller inspected at each times $-1, -2, \ldots$—we can classify each attacker that arrived before time 0 into several types depending on whether he is detected. Specifically, we call it a type k attacker if he was detected at time $-k$, $k = 1, 2, \ldots$; a type 0 attacker if his attack completed before time 0; or a type i' attacker if the attack is still ongoing at node i at time 0, $i = 1, 2, \ldots, n$.

Because each attacker that has arrived belongs to each type with some probability based on his arrival time, independent of the other attackers, it follows from the Poisson sampling theorem that (see, for example, Proposition 5.3 in [21]) the numbers of different types of attackers are independent Poisson random variables. In addition, for each ongoing attack at time 0, its additional time until completion is also independent of what happens to the other attackers. Consequently, knowing the past inspection results does not provide additional information about the number of ongoing attacks at each node and their additional attack times, beyond what the patroller can glean from the patrol history. Hence, the optimal patrol policy depends only on the patrol history in $[-B, 0)$. $\square$

The preceding theorem allows us to define the state of the system by $\boldsymbol{s} = (s_1, s_2, \ldots, s_{B-1})$, where s_k denotes the node the patroller inspected at time $-k$, for $k = 1, \ldots, B-1$. We write the state space as

$$\Omega = \{(s_1, \ldots, s_{B-1}) : s_k = 1, 2, \ldots, n, \text{ for } k = 1, \ldots, B-1\}. \tag{1}$$

The size of the state space is $|\Omega| = n^{B-1}$ for complete graphs. For other graph types, the state space is smaller because not all states are feasible, with the size being the smallest for line graphs.

The current node of the patroller is indicated by s_1. For any given state, the future of the process is independent from its past, and thus we can formulate the problem as a Markov decision process (MDP). At the end of a time period, the patroller needs to decide whether to stay at the same node for another time period, or move to one of the adjacent nodes. Thus, the action space is $A = \{1, \ldots, n\}$. A deterministic, stationary patrol policy can be delineated by a map π from the state space to the action space $\pi : \Omega \to A$.

Let $a_{i,j} = 1$ if nodes i and j are connected by an edge, or $a_{i,j} = 0$ otherwise, for $i, j = 1, \ldots, n$. Because the patroller can only move to a node adjacent to the current node, a specific mapping $\boldsymbol{s} \to i$ is feasible if and only if $a_{s_1,i} = 1$. We use

$$\mathcal{A}(\boldsymbol{s}) = \{i : a_{s_1,i} = 1\}$$

to denote the set of feasible actions—or equivalently, the set of nodes the patroller can move to—when the process is in state $\boldsymbol{s}$.

The transition probability of this MDP is deterministic. If the patroller next goes to node $i \in \mathcal{A}(\boldsymbol{s})$ when in state $\boldsymbol{s}$, the system will transition to state $\tilde{\boldsymbol{s}} = (\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_{B-1})$, with

$$\tilde{s}_k = \begin{cases} s_{k-1}, & \text{if } k > 1 \ , \\ i, & \text{if } k = 1 \ . \end{cases}$$

For notational simplicity, we write $\phi(\boldsymbol{s}, i)$ for the resulting state if the patroller goes to node i in state $\boldsymbol{s}$. Namely, $\phi(\boldsymbol{s}, i) = \tilde{\boldsymbol{s}}$.

We next consider the cost function for this MDP. Recall that, for a state-action pair $(\boldsymbol{s}, i)$, the patroller completes inspection at node i at time 0. To determine the expected cost incurred in the time interval $[0, 1]$, for $j = 1, \ldots, n$ and $k = 1, \ldots, B-1$, define

$$v_{jk} = \begin{cases} 1, & \text{if } s_k = j, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

In other words, $v_{jk} = 1$ indicates that the patroller inspected node j at time $-k$. For instance, if $n = 3$, $B = 5$, and the current state is $\boldsymbol{s} = (2, 1, 2, 3)$, then

$$[v_{jk}] = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

To compute the expected number of attacks that complete at node $j \neq i$ in $[0, 1]$, first consider an attack that initiates at time $t \in [0, 1]$. Such an attack will complete before time 1 with probability $F_j(1-t)$; it is impossible to detect this attack no matter what the patroller does. Next, consider an attack that initiated at time $-t$, for $t \in [m, m+1]$. This attack will complete in $[0,1]$ if (1) its attack time lies in $[t, t+1]$, and (2) it evades detection at times $-1, -2, \ldots, -m$, which occurs with probability

$$(F_j(t+1) - F_j(t))(1-\alpha_j)^{\sum_{k=1}^{m} v_{jk}}.$$

The preceding argument holds true for $m = 0, 1, \ldots, B-1$. Letting $\lambda_j = p_j \Lambda$—the rate at which attackers arrive at node j, for $j = 1, \ldots, n$—the expected cost due to attack completions at node $j \neq i$ in $[0,1]$ is

$$C_j(\boldsymbol{s}, i) = c_j \lambda_j \left(\int_0^1 F_j(1-t)dt + \sum_{m=0}^{B-1} (1-\alpha_j)^{\sum_{k=1}^{m} v_{jk}} \int_m^{m+1} (F_j(t+1) - F_j(t))dt \right). \quad (3)$$

Because the patroller inspects node i at time 0, the expected cost due to attack completions at node i in $[0,1]$ is

$$C_i(\boldsymbol{s}, i) = c_i \lambda_i \left(\int_0^1 F_i(1-t)dt + \sum_{m=0}^{B-1} (1-\alpha_i)^{1+\sum_{k=1}^{m} v_{ik}} \int_m^{m+1} (F_i(t+1) - F_i(t))dt \right), \quad (4)$$

with the only difference between equations (3) and (4) being the exponent on the $(1-\alpha_i)$ term. Consequently, the cost function for the state-action pair $(\boldsymbol{s}, i)$ for this MDP is

$$C(\boldsymbol{s}, i) = \sum_{j=1}^{n} C_j(\boldsymbol{s}, i).$$

The objective of this MDP is to minimize the total long-run cost rate among the n nodes. Since both the state space and the action space are finite, it follows from Theorem 9.1.8 in Puterman [20] that it is sufficient to consider deterministic, stationary policies. Because the state transition is deterministic, we can define $\psi_\pi(\boldsymbol{s}) \equiv \phi(\boldsymbol{s}, \pi(\boldsymbol{s}))$ as the resulting state, if the patroller applies policy π to state $\boldsymbol{s}$. For an initial state $\boldsymbol{s}_0$, policy π will induce an indefinite, deterministic sequence of states, written by $\{\psi_\pi^k(\boldsymbol{s}_0),\ k = 0, 1, 2, \ldots\}$, where $\psi_\pi^k = \psi_\pi \circ \psi_\pi^{k-1}$, for $k \geq 1$. Because the state space is finite, eventually some state will repeat, and a cycle will continue indefinitely. Therefore, we can write

$$V_i(\pi, \boldsymbol{s}_0) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} C_i(\psi_\pi^k(\boldsymbol{s}_0), \pi(\psi_\pi^k(\boldsymbol{s}_0)))$$

for the long-run cost rate incurred at node i if the patroller applies policy π to initial state $\boldsymbol{s}_0$, which is also equal to the total expected cost due to attacks on node i incurred in a cycle divided by the cycle length. Furthermore, we call the sequence of nodes corresponding to a cycle a *patrol pattern*.

We seek to determine the optimal long-run cost rate over all nodes, namely

$$C^{\mathrm{OPT}}(\boldsymbol{s}_0) \equiv \min_{\pi \in \Pi} \sum_{i=1}^{n} V_i(\pi, \boldsymbol{s}_0), \quad (5)$$

where Π denotes the class of deterministic, stationary patrol policies. Dividing (5) by Λ gives us the minimized long-run average cost incurred for each attack. When $c_i = 1$ for all i, the ratio can be interpreted as the probability of not detecting an attack in time.

While $V_i(\pi, \boldsymbol{s}_0)$ does depend upon $\boldsymbol{s}_0$, the optimal cost rate $C^{\text{OPT}}(\boldsymbol{s}_0)$ does not, if the graph is connected, because $V_i(\pi, \boldsymbol{s}_0)$ depends entirely on the patrol pattern generated by $\boldsymbol{s}_0$ and π. In the rest of this paper, we assume the graph is connected, and write C^{OPT} instead of $C^{\text{OPT}}(\boldsymbol{s}_0)$. To determine the optimal policy, it is equivalent to find the optimal patrol pattern.

2.2 The Optimal Solution

Our MDP model belongs to the class of multichain models described in Chapter 9 of [20], because for a given stationary, deterministic policy, it is possible for the resulting Markov chain to have multiple recurrent classes. To solve for C^{OPT}, we need to solve the following system of equations for $g(\boldsymbol{s})$ and $h(\boldsymbol{s})$ (referred to as the multichain optimality equations in Equations (9.1.1) and (9.1.2) in [20]):

$$g(\boldsymbol{s}) = \min_{i \in \mathcal{A}(\boldsymbol{s})} \{g(\phi(\boldsymbol{s}, i))\}, \qquad \forall \boldsymbol{s} \in \Omega,$$

$$g(\boldsymbol{s}) + h(\boldsymbol{s}) = \min_{i \in \mathcal{B}(\boldsymbol{s})} \{C(\boldsymbol{s}, i) + h(\phi(\boldsymbol{s}, i))\}, \qquad \forall \boldsymbol{s} \in \Omega,$$

where $\mathcal{B}(\boldsymbol{s}) = \{i \in \mathcal{A}(\boldsymbol{s}) : g(\boldsymbol{s}) = g(\phi(\boldsymbol{s}, i))\}$. That is, $\mathcal{B}(\boldsymbol{s})$ is the subset of $\mathcal{A}(\boldsymbol{s})$, including all actions that attain minimum in the first equation. The quantity $g(\boldsymbol{s})$ represents the long-run cost rate if the system starts in state s and $h(\boldsymbol{s})$ is a bias term that can be interpreted as a transient cost. For our system, the optimality equations will have $C^{\text{OPT}} = g(\boldsymbol{s})$ for all $\boldsymbol{s} \in \Omega$, because the long-run cost rate is independent of the initial state. Consequently, in our model we have $\mathcal{B}(\boldsymbol{s}) = \mathcal{A}(\boldsymbol{s})$. As the MDP has a finite state space, we can formulate the following linear program to compute the optimal cost rate C^{OPT} (see Section 9.1.1 in [20] for more details):

$$\max_{g,h} \quad g \tag{6}$$

$$\text{subject to} \quad g + h(\boldsymbol{s}) \le C(\boldsymbol{s}, i) + h(\phi(\boldsymbol{s}, i)), \qquad \forall \boldsymbol{s} \in \Omega \text{ and } i \in \mathcal{A}(\boldsymbol{s}). \tag{7}$$

The size of the constraint matrix is on the order of $|\Omega| n \times |\Omega|$, with the exact number of rows depending on the adjacency structure of the graph. While in principle the linear programming formulation allows us to compute the optimal solution, the method quickly becomes computationally intractable for problems of more than a handful of nodes. For instance, for complete graphs, $|\Omega| = n^{B-1}$, and if we let $n = 7$ and $B = 7$, then the size of the constraint matrix is $7^7 \times 7^6$. The computational intractability motivates the need to develop efficient heuristics to solve the patrol problem.

3 Index Policies Derived from Lagrangian Relaxation

Recall that we seek to determine the minimized long-run cost rate defined in (5), written succinctly as C^{OPT}, because the graph is connected. First, we relax the problem by extending the class of policies so that the patroller is allowed to inspect *any node* at *any real-time point*, as long as the overall long-run inspection rate is no greater than 1. By any real-time point we mean that the detection opportunities do not need to coincide with integers. Whenever the patroller inspects node i, he detects each ongoing attack independently with probability α_i, $i = 1, \dots, n$.

In this relaxed problem, denote by μ_i, $i = 1, \dots, n$, the long-run patrol rate at node i. Let $C_i(\mu_i)$ denote the minimized long-run cost rate, if node i receives a patrol rate μ_i, which will be studied closely in Section 3.1. Write $\boldsymbol{\mu} = (\mu_1, \dots, \mu_n)$ and define

$$\Gamma_1 \equiv \left\{ \boldsymbol{\mu} : \sum_{i=1}^{n} \mu_i \le 1; \mu_i \ge 0, \forall i \right\},$$

and let

$$C^{\mathrm{TR}} \equiv \min_{\boldsymbol{\mu} \in \Gamma_1} \sum_{i=1}^{n} C_i(\mu_i)$$

denote the optimal total cost rate over all nodes, such that each node receives a nonnegative patrol rate, with the sum no greater than 1. It follows immediately that $C^{\mathrm{OPT}} \ge C^{\mathrm{TR}}$, because any policy π induces a set of feasible patrol rates in Γ_1.

We next relax the problem again by incorporating the total-rate constraint $\sum_{i=1}^{n} \mu_i \le 1$ into the objective function with a Lagrange multiplier $w \ge 0$. Define

$$\Gamma_2 \equiv \{ \boldsymbol{\mu} : \mu_i \ge 0, \forall i \},$$

so the difference between Γ_1 and Γ_2 is that $\sum_{i=1}^{n} \mu_i$ needs to be no greater than 1 in Γ_1, but not necessarily so in Γ_2. Define

$$\begin{aligned} C(w) &\equiv \min_{\boldsymbol{\mu} \in \Gamma_2} \left\{ \sum_{i=1}^{n} C_i(\mu_i) + w \left(\sum_{i=1}^{n} \mu_i - 1 \right) \right\} \\ &= \min_{\boldsymbol{\mu} \in \Gamma_2} \sum_{i=1}^{n} \{ C_i(\mu_i) + w\mu_i \} - w. \end{aligned} \tag{8}$$

By incorporating a Lagrange multiplier, we can drop the total-rate constraint, so that in (8), the patroller can inspect any node at any real-time point, by paying a service charge $w > 0$, if he chooses to do so. For any $w > 0$, we have that

$$\begin{aligned} C^{\mathrm{TR}} = \min_{\boldsymbol{\mu} \in \Gamma_1} \sum_{i=1}^{n} C_i(\mu_i) &\ge \min_{\boldsymbol{\mu} \in \Gamma_1} \left\{ \sum_{i=1}^{n} C_i(\mu_i) + w \left(\sum_{i=1}^{n} \mu_i - 1 \right) \right\} \\ &\ge \min_{\boldsymbol{\mu} \in \Gamma_2} \left\{ \sum_{i=1}^{n} C_i(\mu_i) + w \left(\sum_{i=1}^{n} \mu_i - 1 \right) \right\} = C(w). \end{aligned}$$

The first inequality follows because $w > 0$, and $\sum_{i=1}^{n} \mu_i - 1 \le 0$ for any $\boldsymbol{\mu} \in \Gamma_1$; the second inequality follows because $\Gamma_1 \subset \Gamma_2$. Consequently, we have a string of inequalities:

$$C^{\mathrm{OPT}} \ge C^{\mathrm{TR}} \ge C(w).$$

The optimization problem in (8) breaks up the original problem into n separate problems, each concerning a single node. The problem concerning node i can be written as

$$\min_{\mu_i \geq 0} \; C_i(\mu_i) + w\mu_i, \tag{9}$$

where w can be interpreted as the service charge for each inspection at node i. Solving the problem defined by (9) is the first step towards constructing an index policy.

3.1 Single-Node Problem

This section solves the optimization problem in (9), which concerns a single node. We drop the subscript i for notational simplicity. Attackers arrive at a node according to a Poisson process with rate λ, with each taking a random time to complete an attack, according to a distribution function $F(\cdot)$. The node can pay w to receive a patrol inspection at any real-time point, which will detect each ongoing attack with probability α, independent of everything else. A detected attack is removed; an attack that completes before getting detected costs c. The node wishes to minimize the long-run cost rate, which includes cost due to not detecting an attack and the service cost paid to the patroller.

The next theorem shows that, for a given service rate μ, to maximize the long-run rate of detection, it is optimal for the patrol to arrive at intervals of $1/\mu$.

Theorem 2 Consider the optimization problem facing a single node posed in the beginning of Section 3.1. Suppose that the patroller inspects the node at a long-run rate μ, in the sense that the patroller repeats a patrol cycle of length l consisting of m inspections indefinitely, for some positive integer m, such that $\mu = m/l$. To maximize the long-run detection rate, it is optimal to space these m inspections with equal intervals. In other words, it is optimal to inspect the node once every $1/\mu$ time units.

Proof. Let $\boldsymbol{x} = (x_1, \ldots, x_m)$, with x_i denoting the time of the i^{th} inspection in a patrol cycle, $i = 1, \ldots, m$. Without loss of generality, let $x_m = l$. We say that an inspection occurring at time x_i in a patrol cycle is a class-i inspection. Because the patroller repeats the patrol cycle indefinitely, patrol inspections in the same class are l time units apart.

Consider an attack that takes $t < l$ time units to complete, and let N denote the number of patrol inspections during his attack time t. First, because $t < l$, this attack will not see two patrol inspections in the same class. Second, because Poisson arrivals see time average, this attacker will see a class-i inspection with the same probability t/l, for $i = 1, \ldots, m$, so $E[N] = m(t/l)$. Although the distribution of N depends on $\boldsymbol{x}$, its expected value $E[N] = m(t/l)$ does not depend on $\boldsymbol{x}$.

We next determine the policy that maximizes the probability of detecting an attack that takes t time units to complete. Namely, choose $\boldsymbol{x}$ to maximize

$$E[1 - (1 - \alpha)^N],$$

where α is the detection probability of each patrol inspection. By conditioning on N, we can compute

$$\begin{aligned}
E[1-(1-\alpha)^N] &= \sum_{n=0}^{\infty}((1-(1-\alpha)^n)P(N=n)) \\
&= \sum_{n=1}^{\infty}\left(\alpha\sum_{k=0}^{n-1}(1-\alpha)^k P(N=n)\right) \\
&= \alpha\sum_{k=0}^{\infty}\sum_{n=k+1}^{\infty}(1-\alpha)^k P(N=n) \\
&= \alpha\sum_{k=0}^{\infty}(1-\alpha)^k P(N>k).
\end{aligned}$$

Recall that $\sum_{k=0}^{\infty} P(N>k) = E[N] = m(t/l)$, which remains a constant regardless of $\boldsymbol{x}$. Treating $P(N>k)$ as decision variables and replace them with y_k, for all k, the optimization problem can be rewritten as

$$\begin{aligned}
\max \quad & r\sum_{k=0}^{\infty}(1-\alpha)^k y_k, \\
\text{subject to} \quad & \sum_{k=0}^{\infty} y_k = E[N] = \frac{mt}{l} \quad \text{(a constant)}, \\
& 1 \geq y_0 \geq y_1 \geq \cdots \geq 0.
\end{aligned}$$

Because $(1-\alpha)^k$ decreases in k, the optimal solution to the preceding problem is to let

$$y_k = \begin{cases} 1, & k = 0,1,\ldots,\lfloor mt/l \rfloor - 1, \\ mt/l - \lfloor mt/l \rfloor, & k = \lfloor mt/l \rfloor, \\ 0, & k \geq \lfloor mt/l \rfloor + 1. \end{cases}$$

In other words, the optimal choice is for N to take on the two integers surrounding $E[N]$, or just $E[N]$ if it happens to be an integer. This distribution of N can be achieved by setting $x_i = i(l/m)$, or equivalently, by spacing the m patrol inspections with equal intervals of length $1/\mu$. Such a patrol cycle maximizes the probability of detecting an attack that takes $t < l$ time units to complete.

Next, consider the case $t \geq l$, and let $a = \lfloor t/l \rfloor$ and $t' = t - a \times l$. For this attacker, the number of patrol inspections that he sees is at least $a \times m$ over the a complete patrol cycles, with the extra number being the number of patrol inspections covered by length $t' < l$. The same argument shows that it is optimal to space the m patrol inspections with equal intervals of length $1/\mu$.

If the attack time is a random variable, denoted by X, the preceding argument shows that for all t,

$$P\{\text{Detecting the attacker}|X = t\}$$

is maximized with the equal-space patrol policy. Hence,

$$P\{\text{Detecting the attacker}\} = E[P\{\text{Detecting the attacker}|X\}]$$

is also maximized with the equal-space strategy, which completes our proof. □

We next derive an expression for the objection function $C(\mu) + w\mu$, as in (9) with the subscript i stripped off. According to Theorem 2, $C(\mu)$ is simply the long-run cost rate due to attack completions, when patrols occur at fixed intervals $1/\mu$. With the original model setup, each attacker is detected independently with probability α, so an inspection does not constitute a regenerative point of the process. Without regenerative points, it is not possible to use the renewal reward theorem to compute the long-run cost rate.

We consider a variation of the model, where each patrol inspection either detects *all* ongoing attacks with probability α, or detects *none at all* with probability $1 - \alpha$ (instead of detecting each ongoing attack independently with probability α). Because the probability that each attack will be detected remains the same in this model variation, the long-run cost rate remains the same. This variation, however, allows us to define a *renewal* whenever a detection occurs, which makes it possible to compute the long-run cost rate.

From Theorem 2, we only need to consider policies that inspect the node once every y time units, for some real number $y > 0$. A renewal occurs when an inspection results in a detection (detecting all ongoing attacks). Immediately after a renewal, let N denote the number of inspections until the next detection (renewal), which follows a geometric distribution with parameter α. The expected cycle time is

$$E[\text{cycle}] = E[Ny] = yE[N] = \frac{y}{\alpha}.$$

To compute the expected cost in a cycle, note that conditional on $N = n$, then an attack that initiates at time s in the cycle will succeed if its attack time is no greater than $ny - s$, which occurs with probability $F(ny - s)$. Hence (see, for example, Proposition 5.3 in [21]),

$$E[\text{cost}|N = n] = \lambda c \int_0^{ny} F(ny - s)ds + nw = \lambda c \int_0^{ny} F(s)ds + nw.$$

For $t \geq 0$, write

$$\Psi(t) \equiv \int_0^t F(s)ds,$$

for convenience. The expected cost in a cycle is

$$\begin{aligned} E[\text{cost}] &= \lambda c E\left[\int_0^{Ny} F(s)ds\right] + wE[N] \\ &= \lambda c \alpha \left(\sum_{n=1}^{\infty} (1-\alpha)^{n-1}\Psi(ny)\right) + \frac{w}{\alpha}. \end{aligned}$$

Using the renewal-reward theory, the long-run cost rate is therefore

$$\Theta(w, y) \equiv \frac{E[\text{cost}]}{E[\text{cycle}]} = \frac{\lambda c \alpha^2 (\sum_{n=1}^{\infty}(1-\alpha)^{n-1}\Psi(ny)) + w}{y}, \tag{10}$$

if the patroller inspects the node once every y time units, with each inspection costing w.

For a given service charge w, we want to choose the optimal patrol interval y that minimizes $\Theta(w, y)$. For a given service charge w, define

$$g(w) \equiv \max\{y : \Theta(w, y) = \min_x \Theta(w, x)\} \tag{11}$$

as the largest optimal service interval. In other words, we break the tie by choosing the largest service interval at which the optimum occurs.

Theorem 3 The function $g(w)$ defined in (11) increases weakly in w.

Proof. For a given x, the function $\Theta(w, x)$ is linear and increasing in w. Because $\gamma(w) \equiv \min_x \Theta(w, x)$ is the lower envelope of a collection of linear and increasing functions, it follows that $\gamma(w)$ is concave and increasing in w. Furthermore, the right gradient of $\gamma(w)$ is simply $1/g(w)$. Since $\gamma(w)$ is concave, $1/g(w)$ decreases weakly in w, and the result follows. □

Because $g(w)$ increases weakly in w, we can define

$$g^{-1}(y) = \inf\{w : g(w) \geq y\}. \tag{12}$$

The function $g^{-1}(y)$ corresponds to the service charge, for which the patrol interval y is optimal. To compute $g^{-1}(y)$, take the derivative of $\Theta(w, y)$ with respect to y to get

$$\frac{\partial\Theta(w, y)}{\partial y} = \frac{\lambda c\alpha E\left[NF(yN)\right] y - (\lambda c\alpha E[\int_0^{yN} F(s)ds] + w)}{y^2}. \tag{13}$$

Setting the derivative to 0 and solving for w yields

$$\begin{aligned} g^{-1}(y) &= \lambda c\alpha \cdot E\left[yNF(yN) - \int_0^{yN} F(s)ds\right] \\ &= \lambda c\alpha \cdot E\left[\int_0^{yN} (F(yN) - F(s))ds\right] \\ &= \lambda c\alpha^2 \sum_{n=1}^{\infty} (1-\alpha)^{n-1} \left(ny \cdot F(ny) - \Psi(ny)\right). \end{aligned} \tag{14}$$

If $y > B$, the preceding simplifies to $g^{-1}(y) = \lambda c\alpha E[X]$, which is also the expected cost that can be saved by a patrol inspection, if the last one was longer than B time units ago.

Theorem 4 The function $g^{-1}(y)$ defined in (14) increases for $y < B$. In addition, in the single-node problem, if $w = g^{-1}(y)$ for $y < B$, then it is optimal to inspect the node once every y time units. Moreover, if $w \geq g^{-1}(B) = \lambda c\alpha E[X]$, then it is optimal not to inspect the node at all.

Proof. We first rewrite (14) as

$$g^{-1}(y) = \lambda c\alpha^2 \sum_{n=1}^{\infty} (1-\alpha)^{n-1} \left(\int_0^{ny} (F(ny) - F(s))ds\right).$$

The integrand is positive and increases with y and thus the integral and $g^{-1}(y)$ increase with y. If $w = g^{-1}(y)$ for $y < B$, then by construction $\partial\Theta(w, y)/\partial y = 0$. To see $\Theta(w, y)$ is convex in y, compute

$$\frac{\partial^2\Theta(w, y)}{\partial y^2} = \frac{\lambda c\alpha E\left[(2N-1)F(yN)\right]}{y^2} > 0,$$

since N follows a geometric distribution, which is at least 1. If $w = g^{-1}(y)$, then the minimum occurs at y, so it is optimal to inspect once every y time periods. Finally, by convexity we know that the derivative in (13) is increasing. For any $y > B$, we can simplify equation (13) to

$$\frac{\partial\Theta(w, y)}{\partial y} = \frac{\lambda c\alpha E[X] - w}{y^2}.$$

Consequently, if $w \geq g^{-1}(B) = \lambda c\alpha E[X]$, then $\Theta(w, y)$ is a strictly decreasing function and it is optimal for the patroller to never inspect the node. $\square$

A downside of using (14) to compute $g^{-1}(y)$ is that it involves a sum of infinitely many terms. To transform it to eliminate the infinite sum, define

$$b_k(y) \equiv (ky \cdot F(ky) - \Psi(ky)) - ((k-1)y \cdot F((k-1)y) - \Psi((k-1)y)),$$

for $k = 1, 2, \ldots$, which allows us to rewrite (14) as

$$\begin{aligned} g^{-1}(y) &= \lambda c\alpha^2 \sum_{n=1}^{\infty}\left((1-\alpha)^{n-1}\sum_{k=1}^{n} b_k(y)\right) = \lambda c\alpha^2 \sum_{k=1}^{\infty}\left(b_k(y)\sum_{n=k}^{\infty}(1-\alpha)^{n-1}\right) \\ &= \lambda c\alpha \sum_{k=1}^{\infty} b_k(y)(1-\alpha)^{k-1}. \end{aligned} \tag{15}$$

However, $b_k(y) = 0$ if $(k-1)y > B$, so the preceding is a sum of a *finite* number of terms.

Let's now return to the problem with n nodes. Recall that with the next inspection opportunity occurring at time 0, the state of the MDP described in Section 2.1 can be delineated by $\boldsymbol{s} = (s_1, s_2, \ldots, s_{B-1})$, with s_k indicating the node that was inspected at time $-k$, for $k = 1, \ldots, B-1$. For each *node*, we can write its state as

$$\boldsymbol{v} = (v_1, \ldots, v_{B-1}),$$

where $v_k = 1$ if the node receives a patrol inspection at time $-k$, and $v_k = 0$ otherwise, for $k = 1, \ldots, B-1$. To define an index heuristic, we need to map from a node's state to a real number, namely the index, and let the patroller inspect the adjacent node with the highest index value. The standard method for doing so is to determine the service charge, for which a patrol inspection and the node's state together constitute an optimal policy. Such a construction, however, is not possible in our problem, because there may not exist a service charge for which an arbitrary patrol schedule $\boldsymbol{v}$ is optimal. We next present two approaches to construct indices in Sections 3.2 and 3.3.

3.2 Calibrate with the Number of Ongoing Attacks

This method uses the *expected number of ongoing attacks at time 0* as a surrogate to map from a node's state to an index. For a given state $\boldsymbol{v}$, we first compute the expected number of ongoing attacks at time 0. We then find the corresponding patrol interval y, such that at each inspection the patroller will find the same expected number of ongoing attacks. Finally, we map y to the fair service charge $g^{-1}(y)$ in (15) to obtain the index. We explain the details below.

An attack that initiated at time $-t$ will still be ongoing at time 0, if (1) its attack time is greater than t, and (2) it evades detection during these t time units. Hence, for an attack that initiated at time $-t$, for $t \in [k, k+1]$, the probability that it will still be ongoing at time 0 is $\bar{F}(t)(1-\alpha)^{\sum_{i=1}^{k} v_i}$, for $k = 0, 1, \ldots, B-1$. Therefore, the expected number of ongoing attacks at time 0 is

$$\rho(\boldsymbol{v}) = \sum_{k=0}^{B-1} \lambda \int_{k}^{k+1} \bar{F}(t)(1-\alpha)^{\sum_{i=1}^{k} v_i} dt. \tag{16}$$

The larger this quantity, the more attacks the patroller can potentially detect at this node, and hence the more incentive for the patroller to go there.

To map $\rho(\boldsymbol{v})$ to an index, consider a patrol policy with fixed patrol intervals y. Whenever the patroller inspects the node, the expected number of ongoing attacks is equal to

$$\begin{aligned} h(y) &= \lambda \int_0^B \bar{F}(t)(1-\alpha)^{\lfloor t/y \rfloor} dt \\ &= \lambda \left(\sum_{k=1}^{\lfloor B/y \rfloor} (1-\alpha)^{k-1} \int_{(k-1)y}^{ky} \bar{F}(t) dt + (1-\alpha)^{\lfloor B/y \rfloor} \int_{\lfloor B/y \rfloor \cdot y}^{B} \bar{F}(t) dt \right). \end{aligned} \tag{17}$$

The function $h(y)$ increases weakly in y, and is left continuous (because the floor function is right continuous). We can define an inverse function

$$h^{-1}(\rho) = \max\{y : h(y) \le \rho\}.$$

Consequently, the index for state $\boldsymbol{v}$ is

$$W(\boldsymbol{v}) = g^{-1} \circ h^{-1} \circ \rho(\boldsymbol{v}). \tag{18}$$

3.3 Calibrate with the Number of Ongoing Attacks and Their Near-Future Departure Rates

The downside of the index defined in (18) is that it maps from a node state to an index using only the expected number of ongoing attacks at the node, but not how much longer these attacks will remain there. Intuitively, the sooner those attacks will complete, the more urgent it is to inspect the node, so the index should be adjusted higher. One way to develop an index that takes into account how soon ongoing attacks will complete is to search for the exponential attack time distribution whose rate yields the closest fit to the departure rate of ongoing attacks *in the near future*, and then use the index derived from the model with the exponential attack time distribution.

To begin, consider a single-node model in which the attack time distribution is *exponential* with rate θ. By substituting $F(t) = 1 - e^{-\theta t}$ into (14), we can compute the corresponding fair service charge, in terms of the patrol interval y and the exponential rate θ, by

$$\frac{\lambda c \alpha y}{1-(1-\alpha)e^{-\theta y}} \left(\frac{1-e^{-\theta y}}{\theta y} - \frac{\alpha e^{-\theta y}}{1-(1-\alpha)e^{-\theta y}} \right), \tag{19}$$

where λ is the arrival rate of attackers, c the cost for each completed attack, and α the detection probability. Let ρ denote the expected number of ongoing attacks in this model when inspections occur; substitute $\bar{F}(t) = e^{-\theta t}$ into (17) to get

$$\rho = \frac{\lambda}{\theta} \frac{1-e^{-\theta y}}{1-(1-\alpha)e^{-\theta y}}.$$

Solving for y from the preceding yields

$$y = \frac{1}{\theta} \ln \left(\frac{\lambda - \rho\theta(1-\alpha)}{\lambda - \rho\theta} \right).$$

In order to express the fair charge in (19) in terms of ρ rather than y, substitute the preceding into (19) to arrive at

$$W(\rho, \theta) = \rho c \alpha - \frac{c}{\lambda\theta}(\lambda - \rho\theta)(\lambda - \rho\theta(1-\alpha)) \ln \left(\frac{\lambda - \rho\theta(1-\alpha)}{\lambda - \rho\theta} \right). \tag{20}$$

The preceding is the corresponding index if the expected number of ongoing attacks is ρ, when the attack time distribution is exponential with rate θ.

Now return to the problem with a general attack time distribution $F(\cdot)$. When a node is in state $\boldsymbol{v}$ at time 0, write $\phi_G(\boldsymbol{v}, s)$ for the expected number of ongoing attacks at time s, if there is no inspection over the period $[0, s)$. The ongoing attacks at time s consists of two groups: (1) *old attacks* that are present at time 0, and (2) *new attacks* initiated in the interval $[0, s)$. Therefore,

$$\phi_G(\boldsymbol{v}, s) = \sum_{k=0}^{B-1} \lambda \int_k^{k+1} \bar{F}(t+s)(1-\alpha)^{\sum_{i=1}^k v_i} dt + \lambda \int_0^s \bar{F}(t) dt.$$

where the first term corresponds to old attacks and follows with a similar argument that derives (16).

On the other hand, for a node whose attack time is exponentially distributed with rate θ, if the expected number of ongoing attacks at time 0 is ρ, and if there is no inspection over the period $[0, s)$, then the expected number of ongoing attacks at time s is equal to

$$\phi_E(\rho, \theta, s) = \rho e^{-\theta s} + \lambda \int_0^s e^{-\theta t} dt = \rho e^{-\theta s} + \frac{\lambda}{\theta}(1 - e^{-\theta s}),$$

where the two terms correspond to old attacks and new attacks, respectively.

The idea now is to choose parameters (ρ, θ) to give a good fit between $\phi_G(\boldsymbol{v}, s)$ and $\phi_E(\rho, \theta, s)$ for some region $s \in [0, t)$, so that we can use the index defined in (20) for state $\boldsymbol{v}$. Since the current and the next inspections occur at time 0 and time 1, respectively, we adopt a simple approach by choosing (ρ, θ) to solve the equations

$$\phi_E(\rho, \theta, 0) = \phi_G(\boldsymbol{v}, 0), \tag{21}$$

$$\phi_E(\rho, \theta, 1) = \phi_G(\boldsymbol{v}, 1). \tag{22}$$

To see that there exists a unique solution to these two equations, first note that equation (21) is equivalent to equation (16) and thus fully specifies $\rho(\boldsymbol{v})$. By inspection, $\phi_G(\boldsymbol{v}, 1) \in [0, \rho(\boldsymbol{v}) + \lambda]$ for the value of $\rho(\boldsymbol{v})$ calculated in (21). In addition, $\phi_E(\rho, \theta, 1)$ is continuous in θ, and decreases monotonically from $\rho + \lambda$ to 0 as θ increases from 0 to ∞. Hence, by the intermediate value theroem there exists a unique $\theta(\boldsymbol{v})$ satisfying $\phi_E(\rho(\boldsymbol{v}), \theta(\boldsymbol{v}), 1) = \phi_G(\boldsymbol{v}, 1)$.

Finally, write $(\rho(\boldsymbol{v}), \theta(\boldsymbol{v}))$ for the (ρ, θ) solution to the system of equations defined by (21)–(22). The index for state $\boldsymbol{v}$ is therefore $W(\rho(\boldsymbol{v}), \theta(\boldsymbol{v}))$, as defined in (20).

3.4 Improve the Heuristics by Looking Ahead

In Sections 3.2 and 3.3, we present two methods to compute an index based on a node's state $\boldsymbol{v}$, written by $W(\boldsymbol{v})$. Now return to the patrol problem on a graph with n nodes, and recall the definition of the state of the system from (1), and the definition of the state of a node from (2). For each state of the system $\boldsymbol{s}$, we can extract the state of node i, and determine the corresponding index of node i in that state, $i = 1, \ldots, n$. By affixing a subscript i to indicate node i, we now write $W_i(\boldsymbol{s})$ as the index for node i when the system is in state $\boldsymbol{s}$. A straightforward way to define an index heuristic is for the patroller to go to the adjacent node with the highest index. We call this patrol policy the *index heuristic* (IH).

The IH works well on complete graphs, but not necessarily on less connected graphs. If the patroller moves to a leaf node, whose only adjacent node has an extremely small attacker arrival rate, then the patroller may get stuck at the leaf node. To overcome this downside, we allow the patroller to look ahead a few time periods to compute an aggregate index. Such a computation is possible because the state of each node depends entirely on the patrol path, without involving any randomness. Based on previous work [15], we can interpret the indices of the *unselected* nodes as penalties. With this interpretation, an l-step look-ahead aggregate index of a path is the sum of all indices of unselected nodes accumulated over that path in the next l time periods. The patroller can list all possible paths of length l and choose the next node to inspect based on the smallest aggregate index among all those paths. We call it the *index penalty heuristic with depth d*, or IPH(d), if we compare the d patrol patterns generated by look-ahead windows $l = 1, 2, \ldots, d$, and choose the best one. Even though the IPH(d) computes the aggregate index for an l-step path, for $l = 1, \ldots, d$, the aggregate index is used to determine only the next node. Once at the next node, the same procedure is repeated to determine the next node.

Regardless of the choice of the look-ahead window l, the index policy maps from a state to a node. Because the state transition is deterministic, whenever the process enters the same state, the IPH will generate the same patrol sequence. Therefore, the patrol schedule generated by the IPH produces an indefinite repetition of some finite *patrol pattern.* For a given patrol pattern, we can evaluate its long-run cost rate in a straightforward manner.

4 Index Policies Based on Approximate Dynamic Programming

The second type of heuristic policy presented in this paper is based on approximate dynamic programming. In particular, we assume that *in the future*, node i will be inspected at a rate ν_i, $i = 1, \ldots, n$. By assuming a future patrol rate to each node, for a given system state, we can compute the *benefit*—defined as the expected cost saved—if the patroller next inspects a particular node. The heuristic policy is then for the patroller to inspect the node that yields the highest such reward. The future patrol rates—namely $\nu_1, \ldots, \nu_n$—are input parameters of this method. We will discuss the choice of future patrol rates in Section 5.

For given future patrol rates $\nu_1, \ldots, \nu_n$, we offer two methods to approximate the future patrols. In Section 4.1, we assume the future patrols arrive according to a Poisson process. In Section 4.2, we assume the future patrols arrive at fixed intervals.

4.1 Future Patrols Arrive According to Poisson Processes

Recall that time 0 refers to the time point when the next inspection opportunity occurs. Regardless which node the patroller inspects at time 0, we assume that after time 0 the patroller will inspect node i according to a Poisson process with rate ν_i, $i = 1, \ldots, n$. With this assumption, we can compute the *benefit* (expected cost saved) if the patroller inspects node i at time 0, and the heuristic policy is for the patroller to inspect the node that yields the highest such benefit.

We now focus on a single node, and strip off the subscript i for notional convenience. Consider a node state $\boldsymbol{v} = (v_1, \ldots, v_{B-1})$, such that $v_k = 1$ if a patrol inspection occurred at time $-k$, for $k = 1, \ldots, B-1$. Divide the entire time line into three segments. Segment 1 is $(-\infty, -B]$; segment 2 is $(-B, 0]$; segment 3 is $(0, \infty)$. Classify attackers into 3 different types based on the segment when an attacker arrives. We will compute the expected reward collected if the patroller inspects the node at time 0, compared with the case if the patroller does not inspect the node at time 0. We will do so for each of the three attacker types.

Type 1 attackers arrive before time $-B$. Because an attack can last no longer than B time units, whether a type 1 attacker is detected depends entirely on the patrol schedule before time 0. Type 3 attackers arrive after time 0, so whether a type 3 attacker will be detected depends entirely on the patrol schedule after time 0. Therefore, the patrol decision at time 0 only affects the fate of type 2 attackers.

Type 2 attackers arrive in the interval $(-B, 0]$. Any patrol inspection that takes place in $(-B, B]$ has a chance to detect type 2 attackers. Because we cannot change what happened in $(-B, 0)$, to study the effect of whether there is a patrol at time 0, we only need to examine the patrols in the interval $[0, B]$.

First, suppose that there is no patrol at time 0. Future patrols arrive in the time interval $(0, B]$ according to a Poisson process with rate ν. Suppose that there are ℓ patrol inspections in the interval $(0, B]$, and denote these time points by $0 < s_{(1)} < s_{(2)} < \ldots < s_{(\ell)} < B$. Define $s_{(0)} \equiv 0$ and $s_{(\ell+1)} \equiv B$ for notational convenience. If an attack initiates at time $-t$, for $t \in [k, k+1]$, then it will still be ongoing at time $s \in [s_{(m-1)}, s_{(m)}]$ with probability

$$\bar{F}(t+s)(1-\alpha)^{\sum_{i=1}^{k} v_i + m - 1},$$

because the attack has to last for longer than $t+s$, and it has to evade detections at times $-1, -2, \ldots, -k$ and at times $s_{(1)}, s_{(2)}, \ldots, s_{(m-1)}$. This argument holds true for $k = 0, 1, \ldots, B-1$. Next, for $s \in [0, B]$ and a node state $\boldsymbol{v}$, define

$$\begin{aligned}\Phi(s, \boldsymbol{v}) &\equiv \lambda \left(\sum_{k=0}^{B-1} \int_{k}^{k+1} \bar{F}(t+s)(1-\alpha)^{\sum_{i=1}^{k} v_i} dt \right) \\ &= \lambda \left(\sum_{k=0}^{B-1} (1-\alpha)^{\sum_{i=1}^{k} v_i} \int_{k+s}^{k+1+s} \bar{F}(t) dt \right),\end{aligned}$$

which represents the expected number of type 2 attackers who have not been detected by time 0, and whose attack will not complete by time s. Some fraction of these $\Phi(s, \boldsymbol{v})$ attackers, however, will be detected between time 0 and time s. Therefore, the expected number of ongoing type 2 attacks at time $s_{(m)}$ is equal to

$$(1-\alpha)^{m-1} \Phi(s_{(m)}, \boldsymbol{v}).$$

Consequently, if the patroller does not inspect the node at time 0, the expected total number of type 2 attackers who are detected by the ℓ patrols in the interval $(0, B]$ is

$$\alpha \sum_{m=1}^{\ell} (1-\alpha)^{m-1} \Phi(s_{(m)}, \boldsymbol{v}).$$

The preceding quantity is conditional on ℓ patrols in $(0, B]$ at times $s_{(1)}, s_{(2)}, \ldots, s_{(\ell)}$. Because patrols arrive according to a Poisson process with rate ν in $(0, B]$, the values $s_{(1)}, s_{(2)}, \ldots, s_{(\ell)}$ have the same distribution as the order statistics from ℓ independent uniform random variables over $(0, B]$. In other words, conditional on ℓ patrols in the interval $(0, B]$, the probability density function of the arrival time of the mth patrol, namely $s_{(m)}$, is given by

$$\frac{\ell! t^{m-1} (B-t)^{\ell-m}}{(\ell-1)! m! B^{\ell}}, \quad 0 \le t \le B.$$

Consequently, we can compute the expected number of type 2 attackers that are detected in the interval $(0, B]$ by

$$\begin{aligned}\Psi(\nu, \boldsymbol{v}) &\equiv \sum_{\ell=0}^{\infty} \left(\frac{(\nu B)^{\ell}}{\ell!} \exp(-\nu B) \cdot \int_{0}^{B} \left(\alpha \sum_{m=1}^{\ell} (1-\alpha)^{m-1} \Phi(t, \boldsymbol{v}) \right) \frac{\ell! t^{m-1} (B-t)^{\ell-m}}{(\ell-1)! m! B^{\ell}} dt \right) \\ &= \sum_{\ell=1}^{\infty} \left(\frac{(\nu B)^{\ell}}{\ell!} \exp(-\nu B) \sum_{m=1}^{\ell} \alpha (1-\alpha)^{m-1} \int_{0}^{B} \Phi(t, \boldsymbol{v}) \frac{\ell! t^{m-1} (B-t)^{\ell-m}}{(\ell-1)! m! B^{\ell}} dt \right). \quad (23)\end{aligned}$$

Suppose now that we send the patroller to this node at time 0. The expected reward from the inspection at time 0 and also from inspections at the node conducted after 0, which accrues from detecting type 2 attackers, is given by

$$c(\alpha\Phi(0,\boldsymbol{v})+(1-\alpha)\Psi(\nu,\boldsymbol{v})).$$

Now return to the patrol problem with n nodes, and define the functions $\Phi_i(\cdot,\cdot)$ and $\Psi_i(\cdot,\cdot)$ similarly for node i, $i=1,\ldots,n$. If the patroller does not patrol anywhere at time 0, the total expected reward collected from type 2 attackers across all nodes is $\sum_{i=1}^{n} c_i\Psi_i(\nu_i,\boldsymbol{v}_i)$. If the patroller inspects node j at time 0, then the expected reward collected from type 2 attackers across all nodes is

$$c_j(\alpha_j\Phi_j(0,\boldsymbol{v}_j)+(1-\alpha_j)\Psi_j(\nu_j,\boldsymbol{v_j}))+\sum_{i\neq j} c_i\Psi_i(\nu_i,\boldsymbol{v_i}).$$

Thus, the benefit for choosing node j is

$$c_j\alpha_j(\Phi_j(0,\boldsymbol{v}_j)-\Psi_j(\nu_j,\boldsymbol{v_j})),$$

which is the index for node j. The heuristic is for the patroller to go to the adjacent node that yields the highest such value.

4.2 Future Patrols Arrive at Fixed Intervals

In this section, we assume the future patrols arrive at each node at fixed intervals. For a given system state, the patroller needs to decide which node to inspect at time 0. Suppose the patroller inspects node i at time 0, then we assume that the future patrols at node i occur at times $0, 1/\nu_i, 2/\nu_i, \ldots$; for $j \neq i$, we assume that the future patrols occur at times $1/(2\nu_j), 3/(2\nu_j), 5/(2\nu_j), \ldots$. The rationale of using $1/(2\nu_j)$ as the first patrol time for node j is that, if the patrols occur at fixed intervals $1/\nu_j$, then in equilibrium the time until the first patrol follows the uniform distribution over $[0, 1/\nu_j]$. Hence, we use the expected waiting time, when the patrol process is in equilibrium, to approximate the time of the first patrol. With the preceding assumptions, we can compute the *benefit* (expected cost saved) if the patroller inspects node i at time 0, and the heuristic policy is for the patroller to inspect the node that yields the highest such benefit.

We now focus on a single node, and strip off the subscript i for notional convenience. Consider a node state $\boldsymbol{v}=(v_1,\ldots,v_{B-1})$, such that $v_k=1$ if an inspection occurred at time $-k$, for $k=1,\ldots,B-1$. For a given state $\boldsymbol{v}$, we want to compare two patrol schedules:

1. Inspect the node at times $a+kb$, for $k=0,1,2,\ldots$.

2. Inspect the node at times kb, for $k=0,1,2,\ldots$.

By letting $a=1/(2\nu)$ and $b=1/\nu$, these two patrol schedules correspond to the two patrol policies discussed earlier. We want to compute the benefit of using schedule 2 as opposed to schedule 1 for each node.

Again, divide the entire time line into three segments as the case in Section 4.1. Segment 1 is $(-\infty, -B]$; segment 2 is $(-B, a]$; segment 3 is (a, ∞). Classify attackers into 3 different types based on the segment when an attacker arrives. We will compare the expected reward collected between the two patrol schedules, for each of the three attacker types.

Type 1 attackers arrive before time $-B$. Because an attack can last no longer than B, whether a type 1 attacker gets detected depends entirely on the patrol schedule before time 0. Hence, the expected number of type 1 attackers that get detected are identical for the two patrol schedules.

Type 2 attackers arrive in the interval $(-B, a]$. Any patrol inspection that takes place in $(-B, a+B]$ has a chance to detect type 2 attackers. Because the two patrol schedules are identical before time 0, the difference comes from the patrols that take place in $[0, a+B]$.

With schedule 1, the first patrol occurs at time a. First consider the expected number of ongoing type 2 attacks at time a. If an attack initiates at time t, for $t \in [0, a]$, then it will still be ongoing at time a with probability $\bar{F}(a-t)$. If an attack initiates at time $-t$, for $t \in [k, k+1]$, then it will still be ongoing at time a with probability

$$\bar{F}(t+a)(1-\alpha)^{\sum_{i=1}^{k} v_i},$$

because the attack has to last for longer than $t+a$, and it has to evade detections at times $-1, -2, \ldots, -k$. This argument holds true for $k = 0, 1, \ldots, B-1$. Therefore, the expected number of ongoing type 2 attacks at time a is

$$\lambda \left(\int_0^a \bar{F}(a-t)dt + \sum_{k=0}^{B-1} \int_k^{k+1} \bar{F}(t+a)(1-\alpha)^{\sum_{i=1}^{k} v_i} dt \right).$$

In general, the expected number of ongoing type 2 attacks at time $a+mb$, for $m = 0, \ldots, \lfloor B/b \rfloor$, is given by

$$\begin{aligned} \Phi_1(m) &\equiv \lambda \left(\int_0^a \bar{F}(a+mb-t)(1-\alpha)^m dt + \sum_{k=0}^{B-1} \int_k^{k+1} \bar{F}(t+a+mb)(1-\alpha)^{\sum_{i=1}^{k} v_i + m} dt \right) \\ &= \lambda \left((1-\alpha)^m \int_{mb}^{a+mb} \bar{F}(t)dt + \sum_{k=0}^{B-1} (1-\alpha)^{\sum_{i=1}^{k} v_i + m} \int_{k+a+mb}^{k+1+a+mb} \bar{F}(t)dt \right). \end{aligned}$$

Consequently, with patrol schedule 1, the expected total reward collected from type 2 attackers is

$$c\alpha \sum_{m=0}^{\lfloor B/b \rfloor} \Phi_1(m).$$

Next consider schedule 2. Write $\Phi_2(m)$ for the expected number of ongoing type 2 attacks at time mb, for $m = 0, 1, \ldots, \lfloor (a+B)/b \rfloor$. For $m = 0$, we have

$$\Phi_2(0) \equiv \sum_{k=0}^{B-1} \lambda \int_k^{k+1} \bar{F}(t)(1-\alpha)^{\sum_{i=1}^{k} v_i} dt,$$

as given by (16). For $m = 1, \dots, \lfloor (a+B)/b \rfloor$, we can compute

$$\Phi_2(m) \equiv \lambda \left(\int_0^a \bar{F}(mb-t)(1-\alpha)^{m-1} dt + \sum_{k=0}^{B-1} \int_k^{k+1} \bar{F}(t+mb)(1-\alpha)^{\sum_{i=1}^k v_i + m} dt \right)$$
$$= \lambda \left((1-\alpha)^{m-1} \int_{mb-a}^{mb} \bar{F}(t) dt + \sum_{k=0}^{B-1} (1-\alpha)^{\sum_{i=1}^k v_i + m} \int_{k+mb}^{k+1+mb} \bar{F}(t) dt \right).$$

Consequently, with patrol schedule 2, the expected total reward collected from type 2 attackers is

$$c\alpha \sum_{m=0}^{\lfloor (a+B)/b \rfloor} \Phi_2(m).$$

Type 3 attackers arrive after time a, and we claim that the expected reward collected from type 3 attackers is identical for the two patrol schedules. Divide segment 3 into blocks, each with length b, as follows:

$$(a, a+b], (a+b, a+2b], (a+2b, a+3b], \dots.$$

Consider the block $(a, a+b]$. The probability that an attacker arriving at time t, for $t \in (a, 2a]$, will be detected by schedule 1, is the same as the probability that an attacker arriving at time $t - a + b$ will be detected by schedule 2, because this attacker will see the first patrol after $a + b - t$ time units, and thereafter at fixed intervals b. For the same reason, the probability that an attacker arriving at time t, for $t \in (2a, a+b]$, will be detected by schedule 1, is the same as the probability that an attacker arriving at time $t - a$ will be detected by schedule 2. In other words, between the two patrol schedules, there is a one-to-one correspondence between the time points in the block $(a, a+b]$, such that attackers arriving at matching time points have the same probability of getting detected by their respective patrol schedules. Because attackers arrive according to a Poisson process, which has stationary increments, the expected reward collected from type 3 attackers in the block $(a, a+b]$ is thus the same for the two schedules. A similar argument shows that the expected reward collected from type 3 attackers in each of the blocks in segment 3 is the same for the two schedules.

To sum up, the two patrol schedules collect the same expected reward from types 1 and 3 attackers. Therefore, the improvement of schedule 2 over schedule 1 is the difference of rewards collected from type 2 attackers, namely

$$c\alpha \left(\sum_{m=0}^{\lfloor (a+B)/b \rfloor} \Phi_2(m) - \sum_{m=0}^{\lfloor B/b \rfloor} \Phi_1(m) \right).$$

In the patrol problem with n nodes, we can use the preceding to compute the benefit of inspecting each node at time 0. The heuristic is for the patroller to go to the adjacent node that yields the highest such value.

4.3 Improve the Heuristics by Looking Ahead

Recall from Section 2.1 that $\boldsymbol{s}$ denotes the current system state. Using the method in either Sections 4.1 or 4.2, the patroller compares all feasible nodes, and inspects the node that attains the highest benefit; denote this highest benefit by $U(\boldsymbol{s})$. The function $U(\boldsymbol{s})$ can be interpreted as an approximate quality of state $\boldsymbol{s}$, because it represents the additional benefit in state $\boldsymbol{s}$ compared with the baseline value. Therefore, we can use $U(\boldsymbol{s})$ to approximate the reward-to-go function for state $\boldsymbol{s}$, and formulate a dynamic program as follows:

$$\max_{i \in \mathcal{A}(\boldsymbol{s})} \left\{ R(\boldsymbol{s}, i) + U(\phi(\boldsymbol{s}, i)) \right\}, \tag{24}$$

where $\mathcal{A}(\boldsymbol{s})$ denotes the set of nodes that can be inspected in state $\boldsymbol{s}$, and $\phi(\boldsymbol{s}, i)$ the resulting state for the state-action pair $(\boldsymbol{s}, i)$, as defined in Section 2.1. The reward function $R(\boldsymbol{s}, i)$ represents the expected reward collected for the state-action pair $(\boldsymbol{s}, i)$—which is simply the expected number of ongoing attacks at node i at time 0, multiplied by $c_i \alpha_i$. Hence, using (16), we have that

$$R(\boldsymbol{s}, i) = c_i \alpha_i \sum_{k=0}^{B-1} \lambda_i \int_k^{k+1} \bar{F}_i(t)(1-\alpha_i)^{\sum_{m=1}^{k} v_{im}} dt.$$

where $v_{im} = 1$ if $s_m = i$. With this policy, in state $\boldsymbol{s}$ the patroller next inspects the node that attains the highest value in (24).

By taking the idea one step further, we can look ahead l steps. For each path of length l, we can compute the total expected reward accumulated over the path, and the quality of the resulting state approximated by the $U(\cdot)$ function. Hence, we can compare all feasible paths of length l and choose the one that yields the highest value. The patroller moves to the first node in that winning path, and repeats this procedure to determine the next destination.

5 Numerical Experiments

This section studies the various heuristic methods numerically. We consider 5 graph types.

1. Complete graph: All nodes are connected. A complete graph is suitable in the scenario where a security manager sits in a surveillance room watching real-time video feeds from various cameras. The security manager can watch any feed at any time, which is analogous to a patroller moving to any node directly.

2. Line graph: A line graph is applicable to an air-borne patrol unit responsible for a border or a vessel responsible for a river or a coast line.

3. Circle graph: A circle graph is applicable to a ground unit patrolling the boundary of an area.

4. Random tree: A random tree is generated recursively by connecting a new node randomly to an existing node. It is applicable to a patrol car that is responsible for road segments, or a vessel patrolling a river with branches.

5. Hexagon grid: A hexagon grid is popular in war games [10], and is applicable to a patrol unit that covers an open area. Each hexagon corresponds to a node, and the patroller can move between adjacent hexagons. We label the center node as node 1, and nodes 2–7 in the first layer, and nodes 8–19 in the second layer, and so on. A hexagon grid with n nodes consists of nodes 1 to n with this labeling method.

While our heuristic methods work for any bounded attack time distribution, in order to assess the heuristics we use attack time distributions so that the problem does not become trivial. We do not want the attack time to be too short, in which case it will be very difficult for the patroller to detect an attack. As our research goal is to study the effectiveness of patrol policies, we set the attack time to be at least 1 so that each attack, regardless of its arrival time, can be detected by some feasible patrol schedule. We also do not want the attack time to be too large, in which case the patroller will detect almost everything. For a graph with n nodes, we let the attack time be bounded by $B = n$, which also makes the state space manageable for problems of moderate size.

We allow the attack time at each node to follow one of three distributions: deterministic, uniform, and triangular. In the case of a deterministic attack time, we generate a uniform random variable over $[1, n]$ to be its attack time, where n is the number of nodes. In the case of a uniform attack time distribution, we generate two such uniform random variables to be its minimum and maximum. In the case of a triangular attack time distribution, we generate three such uniform random variables to be its minimum, mode, and maximum.

For cost of a successful attack, we generate c_i according to independent uniform distributions over $[1, 3]$. The importance of the nodes are comparable, with the most important node being at most three times as important as the least important one. For attack probability on each node, we first generate $u_1, \ldots, u_n$ according to independent uniform distributions over $[0, 1]$, and then normalize them to get $\hat{u}_i = u_i / \sum_{j=1}^{n} u_j$. We then let $p_i = (1/n + \hat{u}_i)/2$. In other words, we distribute probability 1/2 evenly among the n nodes, and the other 1/2 randomly among the n nodes. In doing so, we avoid the possibility that a node becomes irrelevant because its attack probability is close to 0. For detection probability, we generate α_i according to independent uniform distribution over $[0.4, 1]$. Finally, recall that the value of Λ is inconsequential in our model.

To implement the index policy based on approximate dynamic programming, we need to have a set of input parameters $(\nu_1, \ldots, \nu_n)$, with ν_i being the future patrol rate at node i, for $i = 1, \ldots, n$. In Appendix A, we discuss two methods to derive lower bounds for the optimal long-run cost rate. One method is based on Lagrangian relaxation, and the other method is based on a linear program. For the index policy based on approximate dynamic programming, we use the patrol rates implied from each of these two lower bounds as input parameters $(\nu_1, \ldots, \nu_n)$. Consequently, we study six index policies, including two based on Lagrangian relaxation (LR), and four based on approximate dynamic programming (ADP). These six heuristics are summarized in Table 1.

Figures 1 and 2 compare the performance of six heuristics in Table 1 on complete graph and line graph, respectively. Complete graph represents the most connected graph type, while the line graph represents the least connected one. Although the other graph types are not shown here, the results are similar, and overall we make the following observations:

Table 1: Summary of six heuristic methods.

Name	Method	Details	Section
LR1	LR	Calibrated by the expected number of ongoing attacks	3.2
LR2	LR	Calibrated by the expected number of ongoing attacks and their near-future departure rates	3.3
AP1	ADP	Future patrols Poisson processes, with rates implied from lower bound based on Lagrangian relaxation	4.1 A.1
AP2	ADP	Future patrols Poisson processes, with rates implied from lower bound based on linear programming	4.1 A.2
AF1	ADP	Future patrols at fixed intervals, with rates implied from lower bound based on Lagrangian relaxation	4.2 A.1
AF2	ADP	Future patrols at fixed intervals, with rates implied from lower bound based on linear programming	4.2 A.2

1. For the same look-ahead window, LR2 generally takes less computation time and produces better results than does LR1.

2. For the same look-ahead window, AP1 generally takes less computation time and produces better results than does AP2. AP1 takes less computation time, mainly because computing the lower bound based on Lagrangian relaxation takes less time than computing the lower bound based on linear programming.

3. For the same look-ahead window, AF2 generally takes more computation time and produces better results than does AF1. Overall, AF2 compares favorably to AF1.

4. For the same look-ahead window, AP takes more computation time than does AF, mainly due to numerically computing (23).

From these figures, it appears that the two best heuristics—based on both performance and computation time—are LR2 and AF2. We propose a *hybrid heuristic*, with which we run both LR2 and AF2, and choose the policy that yields a lower long-run cost rate. As seen in Figures 1 and 2, the hybrid heuristic delivers excellent results.

Also seen from Figures 1 and 2, one can always exert more computational effort to improve the heuristic by increasing d. The marginal benefit of increasing d depends on the graph structure. The less connected the graph (such as a line graph), the more pronounced the benefit when d increases. From our numerical experiments, setting

$$d = 1 + \lfloor \text{average distance between all pairs of nodes} \rfloor,$$

for the hybrid heuristic offers a great balance between computation time and performance. Table 2 reports the results of this method, by comparing them to the optimal policy. For complete graphs, we report the results up to 6 nodes, because the linear program formulated in Section 2.2 becomes computationally intractable for 7-node complete graphs on our

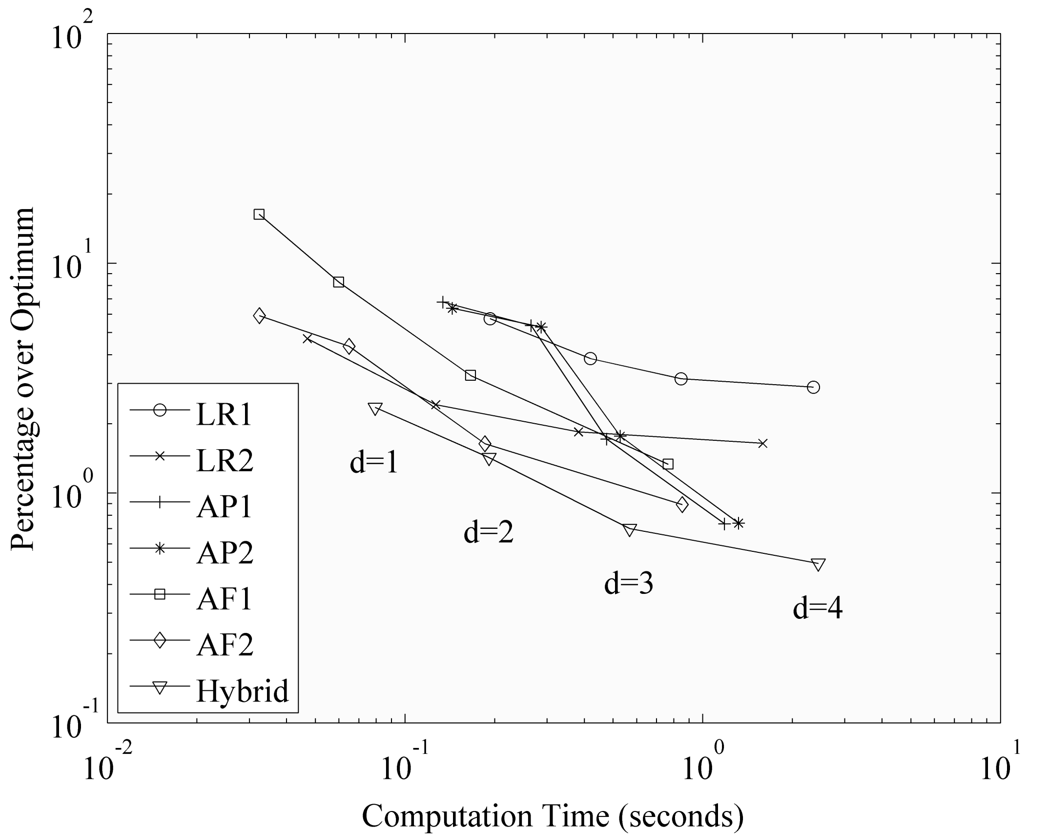

Figure 1: This figure displays the performance of various heuristic methods against computation time for complete graphs with 6 nodes. The performance is the 90th percentile over 1000 random scenarios, reported as percentage over optimum. Each line corresponds to a heuristic method, with $d = 1, 2, 3, 4$ from left to right.

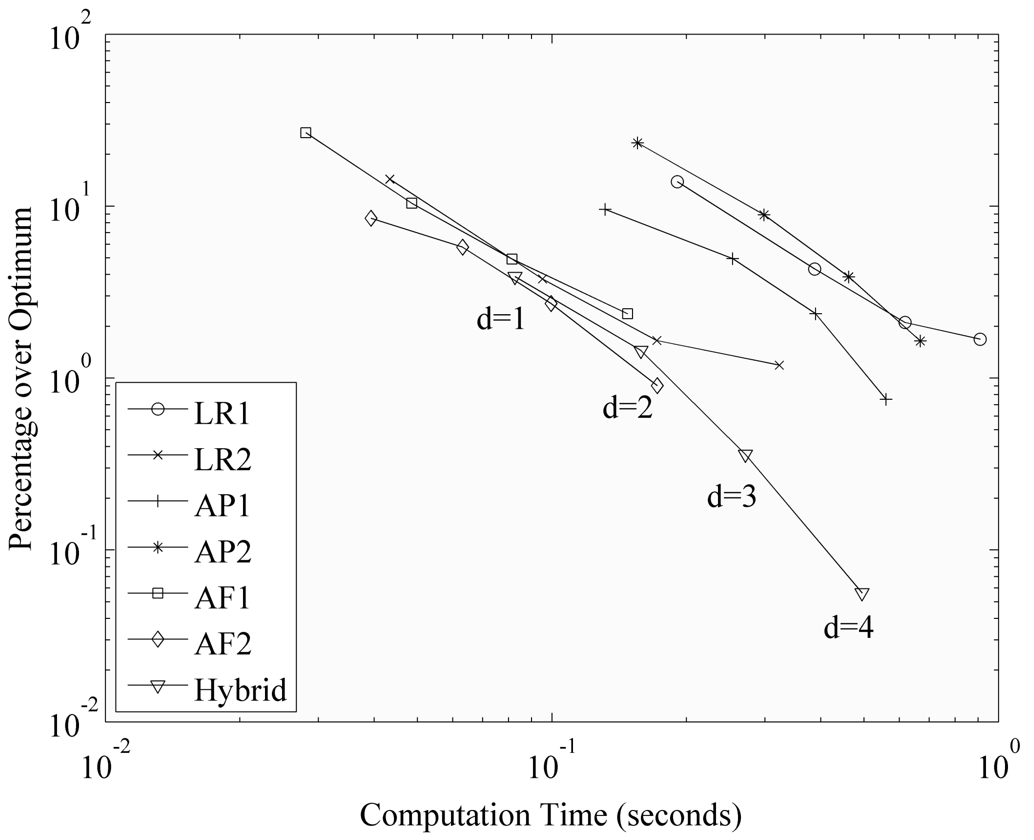

Figure 2: This figure displays the performance of various heuristic methods against computation time for line graphs with 6 nodes. The performance is the 90th percentile over 1000 random scenarios, reported as percentage over optimum. Each line corresponds to a heuristic method, with $d = 1, 2, 3, 4$ from left to right.

computer. We were able to compute the optimal solution on a line graph for up to 9 nodes, because the state space grows slower on a less connected graph. Table 2 shows excellent results for the hybrid heuristic at a fraction of the computation time required for the optimal solution, especially when the graph size grows. Also seen in Table 2 is the quality of the lower bound based on linear programming as discussed in Section A.2. The lower bound does appear to degrade gradually as the graph size grows, faster for less connected graphs and slower for well connected graphs.

Table 2: Performance of the hybrid heuristic, reported as percentage excess over the optimal value; the last column reports the mean of (LB − Opt)/Opt in percentage, where the lower bound is computed using the linear program in Section A.2.

Graph	# Nodes	Mean	50th	75th	90th	Avg Depth	Avg Time Heuristic (sec)	Avg Time Optimal (sec)	Lower Bound
complete	4	0.17	0.00	0.00	0.23	2.0	0.07	0.28	-0.50
complete	5	0.23	0.00	0.00	0.77	2.0	0.13	4.02	-0.52
complete	6	0.42	0.00	0.52	1.43	2.0	0.19	165.81	-0.66
line	4	0.18	0.00	0.00	0.36	2.0	0.07	0.09	-0.46
line	5	0.08	0.00	0.00	0.10	3.0	0.20	0.36	-0.74
line	6	0.12	0.00	0.00	0.36	3.0	0.27	0.64	-1.10
line	7	0.22	0.00	0.06	0.85	3.0	0.55	0.84	-1.56
line	8	0.18	0.00	0.05	0.72	4.0	1.54	4.86	-2.27
line	9	0.27	0.00	0.27	0.89	4.0	2.72	56.67	-2.64
circle	4	0.15	0.00	0.00	0.24	2.0	0.08	0.13	-0.29
circle	5	0.24	0.00	0.00	0.72	2.0	0.12	0.52	-0.48
circle	6	0.49	0.00	0.29	1.70	2.0	0.16	0.77	-0.84
circle	7	0.27	0.00	0.21	0.94	3.0	0.53	1.59	-1.19
circle	8	0.53	0.00	0.57	1.75	3.0	0.95	10.73	-1.76
circle	9	0.69	0.03	0.89	2.15	3.0	1.77	135.68	-2.07
random tree	4	0.14	0.00	0.00	0.16	2.0	0.07	0.09	-0.47
random tree	5	0.16	0.00	0.00	0.36	2.3	0.14	0.39	-0.68
random tree	6	0.19	0.00	0.00	0.78	2.7	0.24	0.69	-1.08
random tree	7	0.20	0.00	0.05	0.69	2.9	0.52	1.43	-1.35
random tree	8	0.31	0.00	0.30	1.05	3.0	0.99	13.78	-1.97
random tree	9	0.31	0.00	0.32	1.12	3.1	1.82	217.87	-2.12
hexagon grid	4	0.15	0.00	0.00	0.24	2.0	0.09	0.20	-0.22
hexagon grid	5	0.24	0.00	0.00	0.78	2.0	0.17	1.30	-0.37
hexagon grid	6	0.38	0.00	0.36	1.39	2.0	0.20	3.85	-0.68
hexagon grid	7	0.50	0.00	0.71	1.67	2.0	0.43	259.63	-0.82

Table 3 compares the hybrid heuristic with the lower bound, for larger graphs. We include the case $n = 6$ so that one can compare the trends in Table 2 and Table 3. As the graph size grows, it takes more computational effort to achieve the same level of performance, and the lower bound degrades gradually, as seen in Table 2. For these two reasons, the gap between the hybrid heuristic and the lower bound widens as the graph size grows, as seen in Table 3. However, from the two columns "Mean" and "Lower Bound" in Table 2, one can see that the hybrid heuristic is closer to the optimal solution, than the optimal solution is to the lower bound. In addition, the gap between the lower bound and the optimal solution appears to widen faster than that between the hybrid heuristic and the optimal solution. Consequently, it is reasonable to conjecture that more than half of the gap between the hybrid heuristic and the lower bound is due to the gap between the optimal solution and the lower bound.

Table 3: Performance of the hybrid heuristic, reported as percentage excess over the lower bound, which is computed using the linear program in Section A.2.

						Avg	Avg Time
Graph	# Nodes	Mean	50th	75th	90th	Depth	(sec)
complete	6	1.12	0.57	1.30	2.45	2.0	0.19
complete	8	1.67	0.93	1.63	3.49	2.0	0.76
complete	10	1.87	1.10	1.94	3.66	2.0	2.74
line	6	1.32	0.07	1.24	3.62	3.0	0.27
line	8	2.66	1.10	3.16	7.22	4.0	1.54
line	10	3.82	2.22	4.80	9.27	4.0	5.49
circle	6	1.42	0.12	1.32	3.94	2.0	0.16
circle	8	2.44	0.98	2.97	6.26	3.0	0.95
circle	10	3.42	2.16	4.23	8.17	3.0	3.72
random tree	6	1.39	0.11	1.14	3.67	2.7	0.24
random tree	8	2.47	1.02	2.63	6.13	3.0	0.99
random tree	10	3.27	1.84	3.97	8.01	3.1	3.96
hexagon grid	6	1.12	0.16	1.20	2.91	2.0	0.20
hexagon grid	8	2.09	1.06	2.37	4.74	2.0	0.75
hexagon grid	10	2.94	1.96	3.71	6.64	2.0	3.38

6 Patrol Against Strategic Attackers

Instead of choosing the attack location according to a probability distribution, as in the model in Section 2, a *strategic attacker* plays a two-person zero-sum game with the patroller. A strategic attacker chooses the attack location in order to maximize the expected cost incurred due to the attack, while the patroller decides how to patrol the graph to minimize it.

One way to formulate this problem is to use the model framework in Section 2, by extending the class of patrol policies to include randomized policies. Let Π^R represent the set of randomized policies—all policies that map from the state space Ω to the action space A according to a probability distribution. For an initial state $\boldsymbol{s}_0$ and a given patrol policy $\pi \in \Pi^R$, the ratio $V_i(\pi, \boldsymbol{s}_0)/\lambda_i$ represents the long-run average cost incurred due to an attack at node i. Hence, the patroller's objective function becomes

$$\min_{\pi \in \Pi^R} \max_{i=1,\ldots,n} \frac{V_i(\pi, \boldsymbol{s}_0)}{\lambda_i}. \tag{25}$$

Because $V_i(\pi, \boldsymbol{s}_0)$ scales proportionally with λ_i, the ratio $V_i(\pi, \boldsymbol{s}_0)/\lambda_i$ does not depend on λ_i. Although $V_i(\pi, \boldsymbol{s}_0)/\lambda_i$ depends on the initial state $\boldsymbol{s}_0$, the optimal solution in (25) does not because we only consider connected graphs. While it is possible to compute the optimal policy via a linear program, as explained in Appendix B.1, that method quickly becomes computationally intractable as the problem size grows.

One way to use the results from the random-attacker case to construct effective patrol policies in the strategic-attacker case is to formulate a two-person zero-sum *matrix game* between the strategic attacker and the patroller. A pure strategy for the strategic attacker is one of the n nodes to attack, while a pure strategy for the patroller is a patrol pattern to be repeated indefinitely. If we can come up with a set of diverse effective patrol patterns for the patroller to choose from, then the value of this matrix game would come close to the optimal value in (25).

To produce effective patrol patterns for this matrix game, we adapt and strengthen the approach in [15]. We generate pure strategies for the patroller—each corresponding to a patrol pattern—using the heuristics against random attackers described in Sections 3 and 4. The set of columns in the matrix game consists of four groups: (1) n patrol patterns, each covering just a single node, (2) 1 patrol pattern designed to provide good coverage to all nodes, (3) n patrol patterns, each designed to cover one node well, and (4) patrol patterns generated by an iterative algorithm.

To generate the patrol patterns in the second and third groups, note that the LR2 index of node i becomes $\alpha_i c_i p_i \Lambda E[X_i]$, if it has been at least B_i time units since the last patrol to node i. By letting $b_i = 1/\alpha_i c_i E[X_i]$ and setting

$$p_i = \frac{b_i}{\sum_{k=1}^n b_k}, \tag{26}$$

each node's index converges to the same value after a long time without patrols. We produce the pattern in the second group by using LR2 for the probabilities defined by (26). For the third group, we modify (26) to create a new probability vector:

$$\begin{aligned} p_i &= \frac{2b_i}{b_i + \sum_{k=1}^n b_k}, \\ p_j &= \frac{b_j}{b_i + \sum_{k=1}^n b_k}, \qquad j \neq i. \end{aligned}$$

The LR2 heuristic for this probability will likely produce a patrol pattern that covers node i well. By repeating the procedure for $i = 1, \ldots, n$, we obtain n patrol patterns that compose the third group.

For the fourth group, we use an approach that is more efficient than the fictitious play method in [15]. The downside of using fictitious play is that in each iteration, the attacker's mixed strategy changes slightly, so that many patrol patterns generated replicate those in earlier iterations. In this paper, we use the following algorithm to generate patrol patterns, which speeds up the process considerably in our numerical experiments.

1. Pick an arbitrary set of patrol patterns $S = \{\xi_1, \dots, \xi_{|S|}\}$. We will add patrol patterns to S to form the fourth group. Initialize $k \leftarrow 0$.
2. If $k = r \times n$, stop, where r is a predetermined positive integer.
3. Construct a two-person zero-sum matrix game with n rows and $|S|$ columns, where row i corresponds to the attacker choosing node i to attack, and column j corresponds to patrol pattern ξ_j. Populate the values in the matrix with the value at (i, j) being the corresponding expected cost incurred, if the attacker attacks node i and the patroller uses patrol pattern ξ_j. Solve the matrix game via linear programming, and write $(p_1, \dots, p_n)$ for the attacker's optimal mixed strategy.
4. Use LR2 to compute ξ, the patrol pattern against the attacker's mixed strategy $(p_1, \dots, p_n)$. If $\xi \in S$, stop; otherwise, update $S \leftarrow S \cup \xi$ and $k \leftarrow k + 1$, and go to Step 2.

When the algorithm stops, we use the patrol patterns in S to form the fourth group. In our numerical experiment, we set $r = 5$ to ensure the algorithm will stop, although in most cases the algorithm stops sooner in Step 4, since no new patrol pattern will be generated. In Step 1, we initialize S to consist of just one patrol pattern, the patrol pattern generated by LR2 by setting $p_i = 1/n$ for all i. It is theoretically possible to further strengthen the fourth group by running the algorithm multiple times with different initial patrol pattern sets, but in practice the improvement is often rather marginal.

When generating patrol patterns in the second, third, and fourth groups, we use LR2 in Section 3.3 with depth set to

$$d = 1 + \lceil (\text{average distance between all pairs of nodes} - 1)/2 \rceil.$$

Finally, we obtain the patroller's heuristic policy by solving the matrix game, where the patroller can use any patrol pattern in all four groups. Table 4 displays the numerical results. The optimal solution and the lower bound used in Table 4 are simple modifications of the corresponding approaches used in the random-attacker case, so we defer the details to Appendix B. As seen in Table 4, our heuristic policy achieves, on average, within 1% of the optimal policy. As the problem size grows, the computation time becomes significantly less than what is required to compute the optimal policy.

7 Conclusions

This paper studies a graph patrol problem, where a patroller traverses a graph through edges to detect potential attacks at nodes. The significance of our work is to allow the possibility

Table 4: Performance of the heuristic in the strategic-attacker case, reported as percentage excess over the optimal value; the last column reports the mean of (LB − Opt)/Opt in percentage, where the lower bound is computed using the linear program in Section B.2.

Graph	# Nodes	Mean	50th	75th	90th	Avg Depth	Avg Time Heuristic (sec)	Avg Time Optimal (sec)	Lower Bound
complete	4	0.11	0.00	0.01	0.26	1.0	0.30	0.15	-0.05
complete	5	0.10	0.00	0.03	0.29	1.0	0.73	2.34	-0.03
complete	6	0.11	0.00	0.02	0.33	1.0	1.77	396.04	-0.02
line	4	0.29	0.05	0.35	0.85	2.0	0.43	0.06	-0.50
line	5	0.37	0.14	0.48	1.01	2.0	0.83	0.21	-0.83
line	6	0.60	0.25	0.78	1.57	2.0	1.66	0.79	-1.24
line	7	0.75	0.36	0.99	1.95	2.0	3.35	3.26	-1.57
line	8	0.99	0.53	1.23	2.53	2.0	6.86	15.16	-1.93
line	9	0.76	0.46	0.96	1.83	3.0	22.49	102.95	-2.15
circle	4	0.27	0.05	0.32	0.79	2.0	0.47	0.08	-0.21
circle	5	0.52	0.23	0.66	1.32	2.0	0.87	0.31	-0.50
circle	6	0.57	0.31	0.75	1.44	2.0	1.68	1.20	-0.86
circle	7	0.77	0.44	1.04	1.92	2.0	3.27	5.29	-1.11
circle	8	0.95	0.58	1.25	2.31	2.0	6.54	26.38	-1.36
circle	9	1.08	0.71	1.36	2.35	2.0	13.42	224.40	-1.55
random tree	4	0.25	0.04	0.29	0.71	2.0	0.40	0.06	-0.47
random tree	5	0.37	0.12	0.48	1.02	2.0	0.84	0.25	-0.82
random tree	6	0.52	0.22	0.69	1.30	2.0	1.69	0.94	-1.17
random tree	7	0.64	0.32	0.84	1.64	2.0	3.42	4.76	-1.39
random tree	8	0.86	0.50	1.11	1.98	2.0	7.01	30.26	-1.69
random tree	9	1.00	0.58	1.27	2.26	2.0	14.53	540.47	-1.93
hexagon grid	4	0.20	0.00	0.16	0.62	2.0	0.54	0.12	-0.12
hexagon grid	5	0.32	0.08	0.43	0.93	2.0	1.13	0.70	-0.29
hexagon grid	6	0.50	0.21	0.62	1.37	2.0	2.25	9.75	-0.46
hexagon grid	7	0.56	0.26	0.75	1.42	2.0	4.76	592.89	-0.48

of overlooking an ongoing attack. The model framework is quite general, allowing for any connected graph topology, attack time distributions, costs, and detection probabilities that vary by node. We primarily focus on the random-attacker case, where the attacker chooses the node to attack according to a known probability distribution. Because computing the optimal solution is only practical for small graphs, our main contribution is to develop index heuristics—based on Lagrangian relaxation, and also on approximate dynamic programming. The novelty of the first method is to use both the *expected number of ongoing attacks* and *their departure rates in the near future* as a conduit to develop an index, while the novelty of the second method is to approximate the value of an inspection using future patrol rates implied by a lower bound for the optimal solution. The heuristics typically achieve within 1% of optimality on a variety of graph topologies. When facing a strategic attacker—who plays a two-person zero-sum game with the patroller—a randomized patrol policy that uses the patrol patterns generated from these index heuristics also performs excellently in our numerical experiments.

In our numerical experiments, we report results for graphs with up to 10 nodes. Such a graph size has applications to practical patrol scenarios; for example, Shieh et al. [22] divided the port of Boston into 9 nodes. For graphs with up to 20 nodes, the optimal policy quickly becomes computationally prohibitive (see Table 4), while our method can still produce effective patrol policies in a timely fashion. For graphs with more than 20 nodes, having just one patroller may not be able to achieve a meaningful level of performance. The coordination among several patrollers presents a significant challenge for patrolling a large area, but it is outside the scope of this paper.

There are a few future research directions that will make the model more applicable to real-world scenarios. For instance, if the locations subject to attacks are dispersed in an area, one may want to incorporate explicit transit times between nodes. The inspection time of each node can be different depending on its size. Our work assumes that the outcome is binary; there is no cost as long as the attack is detected. In some scenarios, such as cyber attacks, the cost may depend on how long it takes for the patroller to detect the attack.

Appendix

A Two Lower Bounds

A.1 Lower Bound Based on Lagrangian Relaxation

This section presents a lower bound for C^{OPT} based on Lagrangian relaxation. Recall from Section 3 that

$$C^{\mathrm{TR}} \equiv \min_{\boldsymbol{\mu}\in\Gamma_1} \sum_{i=1}^{n} C_i(\mu_i) = \max_{w} C(w)$$

is a lower bound for the optimal value. Because $C_i(\mu_i)$ is nonincreasing and convex in μ_i, to compute C^{TR} it is equivalent to solve a classic resource allocation problem; see, for example, [11]. This method does not take into account graph structure, and allows the time between two consecutive patrol inspections to be any real number, as opposed to just integers. Because of these reasons, this lower bound is not expected to be particularly tight. Nevertheless, the patrol rates implied by this lower bound can be used as future patrol rates for the approximate dynamic programming method discussed in Section 4.

A.2 Lower Bound Based on Linear Programming

This section presents a linear program that computes a lower bound for C^{OPT}. The linear program is a slight modification to that used to compute a lower bound in Section 3.3.4 in [15] when the detection probability is 1 for all nodes. Here we summarize the method and highlight the difference.

To begin, consider an arbitrary mixed patrol strategy, and denote by y_{ik} the *expected rate* at which the patroller inspects node i exactly k time units after his previous inspection at node i. We use the term *expected rate* because the patrol strategy may involve randomizing a few patrol patterns. Ideally we would like to find an expression for the expected number of ongoing attacks at node i, if the patroller inspects node i exactly k time units after his previous inspection at node i. Without knowing whether the patroller inspected node i between B_i time units ago and k time units ago, however, it is not possible to do so, as some attackers may have evaded detection during the patroller's last inspection k time units ago. Nevertheless, we can determine an *upper bound* on the expected number of ongoing attacks by assuming that the patroller did not inspect node i between B_i time units ago and k time units ago, which is

$$\lambda_i \int_0^k \bar{F}_i(t)dt + \lambda_i \int_k^{B_i} (1-\alpha_i)\bar{F}_i(t)dt.$$

Hence, each time the patroller inspects node i with the previous inspection being k time periods ago, the expected cost that can be avoided is *at most*

$$R_i(k) \equiv r_i c_i \lambda_i \left(\int_0^k \bar{F}_i(t)dt + \int_k^{B_i} (1-\alpha_i)\bar{F}_i(t)dt \right),$$

for $k = 1, 2, \ldots$.

Because $R_i(k) = c_i\lambda_i E[X_i]$ for $k \geq B_i$, by redefining y_{i,B_i} to be the expected rate at which the patroller inspects node i with the previous inspection *at least* B_i time units ago, we can write

$$\sum_{k=1}^{B_i} y_{ik} R_i(k)$$

as an upper bound for the long-run rate at which the patroller avoids cost at node i. Consequently, the total long-run cost rate is *at least*

$$\sum_{i=1}^{n} \left(c_i\lambda_i - \left(\sum_{k=1}^{B_i} y_{ik} R_i(k) \right) \right). \tag{27}$$

To create a linear program to compute the lower bound, let x_{ij} denote the *expected rate* at which the patroller moves from node i to node j. The linear program goes as follows:

$$\begin{aligned}
\min \quad & \sum_{i=1}^{n}\left(c_i\lambda_i - \left(\sum_{k=1}^{B_i} y_{ik}R_i(k)\right)\right) & \\
\text{subject to} \quad & \sum_{j=1}^{n} x_{ji} = \sum_{j=1}^{n} x_{ij}; \qquad i = 1,2,\ldots,n, & (28)\\
& x_{ij} = 0, \qquad \text{if } a_{ij} = 0; & (29)\\
& \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij} = 1; & (30)\\
& y_{i,1} = x_{ii}, \qquad i = 1,2,\ldots,n; & (31)\\
& \sum_{k=1}^{B_i} y_{ik} = \sum_{j=1}^{n} x_{ji}, \qquad i = 1,2,\ldots,n; & (32)\\
& \sum_{k=1}^{B_i} k y_{ik} \le 1, \qquad i = 1,2,\ldots,n; & (33)\\
& x_{ij}, y_{ik} \ge 0. & (34)
\end{aligned}$$

Constraint (28) ensures flow balance at each node; constraint (29) observes the edge constraint; constraint (30) ensures the total rate to be 1. Constraint (31) follows because either side represents the rate at which the patroller inspects node i in two consecutive time periods. Constraint (32) follows because either side represents the long-run rate at which the patroller enters node i. Constraint (33) is key in tightening the lower bound. The quantity ky_{ik} represents the *expected* long-run fraction of time the patroller spends between two inspections to node i with the two inspections being k time units apart, for $k = 1, \ldots, B_i - 1$. It is an inequality for two reasons: (1) we define y_{i,B_i} as the expected rate at which the patroller enters node i with the previous inspection at least B_i time units ago, and (2) the mixed strategy may include a patrol pattern that never inspects node i.

B The Strategic-Attacker Case

B.1 Optimal Policy for Strategic-Attacker Case

It is possible to compute the optimal value in (25). To do so, it is helpful to first write the dual of the linear program defined by (6)–(7) in Section 2.2.

$$\min_{x} \quad \sum_{\boldsymbol{s}\in\Omega}\sum_{i\in\mathcal{A}(\boldsymbol{s})} C(\boldsymbol{s},i)x(\boldsymbol{s},i) \tag{35}$$

$$\text{subject to} \quad \sum_{i\in\mathcal{A}(\boldsymbol{s})} x(\boldsymbol{s},i) - \sum_{\boldsymbol{t}\in\Omega}\sum_{i\in\mathcal{A}(\boldsymbol{t})} I(\phi(\boldsymbol{t},i)=\boldsymbol{s})x(\boldsymbol{t},i) = 0, \quad \forall \boldsymbol{s}\in\Omega, \tag{36}$$

$$x(\boldsymbol{s},i) \geq 0, \quad \forall \boldsymbol{s}\in\Omega, \text{ and } i\in\mathcal{A}(\boldsymbol{s}). \tag{37}$$

Recall from Section 2.2 that $\phi(\boldsymbol{t},i)$ represents the resulting state if the patroller moves to node i in state $\boldsymbol{t}$. The indicator function $I(\phi(\boldsymbol{t},i)=\boldsymbol{s})$ returns 1, if taking an action i in state $\boldsymbol{t}$ moves the system to state $\boldsymbol{s}$, and returns 0 otherwise. The decision variable $x(\boldsymbol{s},i)$ can be interpreted as the fraction of time the system is in state $\boldsymbol{s}$ with the patroller next moving to node i. Constraint (36) states that the long-run transition rates entering and leaving each state must be the same.

The objective function in (35) calculates the total long-run cost rate over all nodes. For a strategic attacker, however, we want to minimize the largest expected cost per attack among all nodes. The quantity $C(\boldsymbol{s},i)$ aggregates the cost over *all* nodes after the patroller moves to node i and thus does not allow us to isolate the cost at each node. However, $C_i(\boldsymbol{s},j)$, defined in (3) and (4), does specify the cost at the individual node i. The long-run cost rate at node i is therefore

$$\sum_{\boldsymbol{s}\in\Omega}\sum_{j\in\mathcal{A}(\boldsymbol{s})} C_i(\boldsymbol{s},j)x(\boldsymbol{s},j),$$

Dividing the preceding by λ_i, we can redefine the zero-sum game in (25) as

$$\min_{x}\max_{i=1,\dots,n}\sum_{\boldsymbol{s}\in\Omega}\sum_{j\in\mathcal{A}(\boldsymbol{s})}\frac{C_i(\boldsymbol{s},j)}{\lambda_i}x(\boldsymbol{s},j).$$

Therefore, we can modify the linear program in (35)–(37) to minimize the largest long-run average cost per attack among all nodes:

$$\begin{aligned}
\min_{x} \quad & \delta \\
\text{subject to} \quad & \sum_{\boldsymbol{s}\in\Omega}\sum_{j\in\mathcal{A}(\boldsymbol{s})}\frac{C_i(\boldsymbol{s},j)}{\lambda_i}x(\boldsymbol{s},j) \leq \delta, \quad i=1,\dots,n, \\
& \text{and equations (36) to (37).}
\end{aligned}$$

The minimized δ produced by this linear program yields what we want in equation (25). As is the case in Section 2.2, this linear program quickly becomes computationally intractable when the problem size increases.

B.2 Lower Bound for Strategic-Attacker Case

The linear program in Section A.2 can be modified slightly to compute a lower bound for the optimal value for the strategic-attack case. By dividing the cost rate at node i in (27) by λ_i, we see that the expected cost due to an attack at node i is at least

$$c_i - \frac{1}{\lambda_i} \sum_{k=1}^{B_i} y_{ik} R_i(k).$$

Therefore, the optimal solution to the linear program below is a lower bound for the optimal solution to the strategic-attacker case.

$$\begin{aligned} \min \quad & z \\ \text{subject to} \quad & z \geq c_i - \frac{1}{\lambda_i} \sum_{k=1}^{B_i} y_{ik} R_i(k) \\ & \text{constraints (28)–(34)} \end{aligned}$$

Acknowledgments

This material is based upon work supported by the Office of Naval Research. We thank the Associate Editor and two anonymous reviewers for their valuable comments.

References

[1] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-Robot Perimeter Patrol in Adversarial Settings. In *2008 IEEE International Conference on Robotics and Automation, May 19-23, 2008*, pages 1–7, Pasadena, CA, USA, May 2008.

[2] C. Albuquerque, B. J. Vickers, and T. Suda. Network border patrol. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 322–331. IEEE, 2000.

[3] S. Alpern, A. Morton, and K. Papadaki. Patrolling games. *Operations Research*, 59(5):1246–1257, 2011.

[4] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 57–64. International Foundation for Autonomous Agents and Multiagent Systems, 2009.

[5] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending Algorithms for Mobile Robot Patrolling in the Presence of Adversaries to More Realistic Settings. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 557–564. IEEE, Apr. 2009.

[6] N. Basilico, N. Gatti, and F. Villa. Asynchronous Multi-Robot Patrolling Against Intrusions in Arbitrary Topologies. In *The 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1–6, May 2010.

[7] J. Birge and S. Pollock. Modelling rural police patrol. *The Journal of the Operational Research Society*, 40(1):41–54, Jan 1989.

[8] J. Chaiken and P. Dormont. A patrol car allocation model: Capabilities and algorithms. *Management Science*, 24(12):1291–1300, Aug 1978.

[9] K. Chelst. An algorithm for deploying a crime directed (tactical) patrol force. *Management Science*, 24(12):1314–1327, Aug 1978.

[10] J. F. Dunnigan. *The Complete Wargames Handbook: How to Play, Design and Find Them.* Quill, Minneapolis, MN, 1992.

[11] M. S. Kodialam and H. Luss. Algorithms for Separable Nonlinear Resource Allocation Problems. *Operations Research*, 46(2):272–284, Mar. 1998.

[12] B. O. Koopman. The theory of search, part III: The optimum distribution of searching effort. *Operations Research*, 5(5):613–626, 1957.

[13] R. C. Larson. *Urban Police Patrol Analysis.* MIT Press, Boston, 1972.

[14] S. Lee, L. Franz, and A. Wynne. Optimizing state patrol manpower allocation. *The Journal of the Operational Research Society*, 30(10):885–896, Oct 1979.

[15] K. Y. Lin, M. P. Atkinson, T. H. Chung, and K. D. Glazebrook. A graph patrol problem with random attack times. *Operations Research*, 61(3):694–710, 2013.

[16] D. Olson and G. Wright. Models for allocating police preventive patrol effort. *Operational Research Quarterly (1970-1977)*, 26(4):703–715, Nov 1975. Part 1.

[17] D. Portugal and R. P. Rocha. Decision methods for distributed multi-robot patrol. In *Safety, Security, and Rescue Robotics (SSRR), 2012 IEEE International Symposium on*, pages 1–6. IEEE, 2012.

[18] D. Portugal and R. P. Rocha. Distributed multi-robot patrol: A scalable and fault-tolerant framework. *Robotics and Autonomous Systems*, 61(12):1572 – 1587, 2013.

[19] D. Portugal and R. P. Rocha. Scalable, fault-tolerant and distributed multi-robot patrol in real world environments. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 4759–4764. IEEE, 2013.

[20] M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* John Wiley & Sons, New York, NY, 1994.

[21] S. M. Ross. *Introduction to Probability Models.* Academic Press, 10th edition, 2010.

[22] E. Shieh, B. An, R. Yang, M. Tambe, C. Baldwin, J. DiRenzo, B. Maule, and G. Meyer. Protect: A deployed game theoretic system to protect the ports of the united states. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2012.

[23] B. Taylor, L. Moore, E. Clayton, K. Davis, and T. Rakes. An integer nonlinear goal programming model for the deployment of state highway patrol units. *Management Science*, 31(11):1335–1347, Nov 1985.

[24] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated web patrol with strider honeymonkeys. In *Proceedings of the 2006 Network and Distributed System Security Symposium*, pages 35–49, 2006.