OPERATIONS RESEARCH
Vol. 61, No. 3, May–June 2013, pp. 694–710
ISSN 0030-364X (print) | ISSN 1526-5463 (online)

http://dx.doi.org/10.1287/opre.1120.1149
© 2013 INFORMS

A Graph Patrol Problem with Random Attack Times

Kyle Y. Lin, Michael P. Atkinson
Operations Research Department, Naval Postgraduate School, Monterey, California 93943
{kylin@nps.edu, mpatkins@nps.edu}

Timothy H. Chung
Systems Engineering Department, Naval Postgraduate School, Monterey, California 93943, thchung@nps.edu

Kevin D. Glazebrook
Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, United Kingdom,
kevin.glazebrook@ed.ac.uk

This paper presents a patrol problem, where a patroller traverses a graph through edges to detect potential attacks at nodes. To design a patrol policy, the patroller needs to take into account not only the graph structure, but also the different attack time distributions, as well as different costs incurred due to successful attacks, at different nodes. We consider both random attackers and strategic attackers. A random attacker chooses which node to attack according to a probability distribution known to the patroller. A strategic attacker plays a two-person zero-sum game with the patroller. For each case, we give an exact linear program to compute the optimal solution. Because the linear programs quickly become computationally intractable as the problem size grows, we develop index-based heuristics. In the random-attacker case, our heuristic is optimal when there are two nodes, and in a suitably chosen asymptotic regime. In the strategic-attacker case, our heuristic is optimal when there are two nodes if the attack times are deterministic taking integer values. In our numerical experiments, our heuristic typically achieves within 1% of optimality with computation time orders of magnitude less than what is required to compute the optimal policy.

Subject classifications: military: search/surveillance; dynamic programming/optimal control; game/group decisions.
Area of review: Optimization.
History: Received July 2011; revisions received April 2012, September 2012, October 2012; accepted December 2012. Published online in *Articles in Advance* April 12, 2013.

1. Introduction

Patrol problems arise in many real-world situations. Police officers patrol highways and cities; security guards patrol museums and shopping malls; soldiers patrol military bases and borders. In essence, a patrol problem examines how to route the patroller through many locations in order to find illicit activities. With modern technological advancements, the patrol problem can be applied to many more contexts, such as routing unmanned aerial vehicles and speed boats. Whereas in most cases the patroller needs to move physically, it is not always the case. For instance, a high-resolution video camera installed on a surveillance tower or on a blimp can turn to monitor different locations almost instantaneously. A security officer monitoring many locations through real-time video feeds also faces a patrol problem if he can watch only one video feed at a time.

Patrol problems have been studied since the 1970s. Earlier works focused on allocating police patrol resources among different areas to maximize the overall performance (Chaiken and Dormont 1978, Chelst 1978, Larson 1972, Olson and Wright 1975). Besides police patrol in urban areas, there are specialized patrol models for rural areas (Birge and Pollock 1989) and on highways (Lee et al. 1979, Taylor et al. 1985). These earlier works assumed that the frequencies of crimes at different locations remain constant and are known to the patrol force. Game theory has been used to analyze some other problems related to patrol problems, such as search games and infiltration games. In search games, a searcher seeks to find a hider who does not want to be found (Alpern and Gal 2002, 2003; Thomas and Washburn 1991; Zoroa et al. 2009). In infiltration games, an intruder wants to penetrate an area without being caught by the guard (Auger 1991; Baston and Kikuta 2004, 2009; Ruckle 1983; Washburn and Wood 1995).

In this paper, we consider a patrol problem on a graph with n nodes. A patroller needs to traverse the graph through edges to detect potential attacks at nodes. In each time unit, the patroller can move to a node adjacent to his current node and detect any ongoing attacks at the chosen node at the end of that time unit. The probability distribution of the time it takes to complete an attack, as well as the damage an undetected attack causes, depends on the node. There are two common ways to model an attacker's behavior. A *random attacker* chooses which node to attack according to a probability distribution known to the patroller, whereas a *strategic attacker* plays a two-person zero-sum game with the patroller. From a practical standpoint, we are more interested in the strategic attacker case. In this paper, however, we address both cases. The random-attacker case is technically interesting in its own right, and its solution provides valuable insights and paves the way for solving the strategic-attacker case.

There are a few recent studies on patrolling a graph in a game-theoretic setting. Shieh et al. (2012) divided the port of Boston into nine areas (nodes), and formulated the patrol problem as a two-person game. On each day, the defender randomly selects the start time of the patrol, and randomly selects a patrol schedule that needs to leave from and return to the base node. The work closest to our current work is that of Alpern et al. (2011). They studied the case of strategic attackers, and assumed that the time to complete an attack is deterministic and is the same for all nodes. They considered finite-time and infinite-time formulations; in the latter case the patrol must repeat every T time periods for some predetermined T. The optimal solution can only be derived in very special cases. In our paper, we allow each node to have its own attack time distribution and study both random attackers and strategic attackers. In each case, we formulate an exact linear program to compute the optimal solution. Because the linear programs quickly become computationally intractable as the problem size grows, we propose index-based heuristics that are easy to compute (Gittins et al. 2011).

The index-based heuristics have been successful in earlier works (Archibald et al. 2009; Glazebrook et al. 2007, 2009), where a resource (patroller) is dynamically moved among projects (nodes) to optimize system performance. These earlier works focused on the case when the decision maker knows the probability rule that governs the underlying stochastic process—analogous to random attackers. In addition, the earlier works assumed that the resource (patroller) could be moved from one project (node) to any other project instantaneously—analogous to complete graphs. To the best of our knowledge, our work is the first to use index-based heuristics as a vehicle to produce effective policies in a *game-theoretic* setting, and the first to use an *aggregate* index to overcome the constraint on project availability.

The rest of this paper proceeds as follows. Section 2 presents a patrol model. Section 3 discusses the case of random attackers, and §4 discusses the case of strategic attackers. In both cases, we give an exact linear program to compute the optimal solution and propose near-optimal heuristics that are easy to compute. Finally, §5 concludes the paper and points out future research directions.

2. A Patrol Model

In anticipation of an attack, a defender (henceforth the patroller) patrols an area hoping to detect the attack before it completes. An *attack* is broadly construed as an illicit activity undertaken by an adversary, such as breaching a perimeter, surveilling the surroundings, or planting a bomb.

There are n locations in the area subject to attack. To model a patroller's strategy, we embed the n locations in a graph, where each node of the graph represents a location subject to attack. We study the case in which the patroller uses a discrete-time schedule. Two nodes are connected by an edge if the patroller can move from one node to the other in one time period during the patrol. Denote the $n \times n$ adjacency matrix by $\mathbf{a} = \{a_{i,j}\}$, where $a_{i,j} = 1$ if nodes i and j are connected, or $a_{i,j} = 0$ otherwise. By definition, $a_{i,i} = 1$ for all i. In this paper, we only consider connected graphs. A patrol policy is *an indefinite sequence of nodes that observes the edge constraint*.

When an attacker arrives at location i, it takes a random amount of time to complete the attack, called the *attack time* at node i, denoted by X_i, $i = 1, \ldots, n$. The probability distributions of these attack times are arbitrary, but known to both the patroller and the attacker. Whereas an attack can initiate at any real-valued time, the patroller takes one time unit to move from one node to an adjacent node. To facilitate discussions, we assume that the patroller detects an ongoing attack if an attacker and the patroller occupy the same node *at the end of a time period*. An application illustrating this form of detection model can be found in antisubmarine warfare contexts, where a helicopter is equipped with a dipping sonar. The helicopter (patroller) makes discrete observations by lowering the sonar unit into the water, but must retract it prior to transiting to another search location, during which no detections can occur. We assume there are no false negatives. In other words, if the patroller visits node i, then the patroller detects any ongoing attacks at node i at the end of the period. An undetected attack at node i costs c_i to the patroller, $i = 1, \ldots, n$, whereas a detected attack costs 0 regardless of how long the attacker was at the node prior to detection.

Our patrol model has many applications. For instance, the Coast Guard can patrol a port by dividing the port and surrounding waterways into several areas. A security guard can patrol a museum or art gallery. An unmanned aerial vehicle can patrol a combat zone in search of threats. With such applications in mind, the scale of problems appropriate for a single patroller should have a moderate number of nodes, so that there is a reasonable chance of detecting an attack in time. In their work, Shieh et al. (2012) divided the port of Boston into 9 nodes. We suggest that our model is most applicable when there are no more than 20 nodes in the patrol graph. Having a single patroller responsible for a much larger graph will result in a small chance of detecting an attack, thus making the patrol problem both impractical and uninteresting. The one place in the paper where we consider larger graphs (in Theorem 3 and Corollary 1) is also the one place where we consider multiple patrollers.

Loosely speaking, the patroller's objective is to find a patrol policy to minimize the expected cost incurred due to a successful attack, when and if an attack occurs. We consider two versions of the problem—*random attackers* and *strategic attackers*—and discuss them separately in the next two sections.

3. Patrol Against Random Attackers

This section studies the problem in which an attacker will choose node i to attack with probability p_i, $i = 1, \ldots, n$. Whereas we assume the patroller knows p_i, we assume that attacks occur infrequently and the patroller has no knowledge about when an attack will occur. Loosely speaking, the patroller seeks to minimize the expected cost by assuming that an attack will eventually occur after a very long time. The problem starts over after an attack takes place, whether the attack is detected (other security measures ensue) or not (disaster happens). To formulate this objective function, we assume that the attackers arrive according to a Poisson process with rate Λ. Because the Poisson process has stationary and independent increments, this assumption implies that an attack is equally likely to occur at any time moment and that the patroller cannot learn about future attack times from the attack history.

From a practical standpoint, the attack rate Λ is usually extremely small. From the formulation standpoint, however, the value of Λ is inconsequential if we let the problem continue indefinitely by ignoring interruptions from attacks. That is, many attackers can operate simultaneously at the same node, with each acting independently on its own and inflicting damage separately. By minimizing the long-run cost rate in this model, we also minimize the average cost due to each attack, because Λ is just a scaling constant. Consequently, the optimal policy does not depend on the value of Λ.

3.1. MDP Formulation and Optimal Policy

Because each attacker independently chooses to attack node i with probability p_i, attackers arriving at node i constitute independent Poisson processes with respective rates $\lambda_i = p_i\Lambda$, $i = 1, \ldots, n$. If the patroller visits node i in the current period, then according to our assumption, the patroller detects *all* ongoing attacks at node i at the end of the period. Because there are no attackers at node i immediately following a patrol at node i, and the attackers arrive at node i according to a Poisson process, the state of the system can be delineated by $\mathbf{s} = (s_1, s_2, \ldots, s_n)$, where s_i denotes the time periods elapsed since node i was last visited by the patroller, $i = 1, \ldots, n$. The state of each node increments by 1 for each time period without a visit, and returns to 1 immediately after the patroller's visit. We write the state space as

$$\Omega = \{(s_1, \ldots, s_n)\colon s_i = 1, 2, \ldots, \text{ for } i = 1, \ldots, n\}.$$

Because the patroller visits one node in each time period, all s_i, $i = 1, \ldots, n$, have distinct values. In addition, only one s_i has value 1, namely, the node the patroller just visited. Therefore, the current node of the patroller can be represented by $l(\mathbf{s}) = \arg\min_i s_i$.

Because for any given state the future of the process is independent of its past, 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 $A = \{j\colon j = 1, \ldots, n\}$. A deterministic, stationary patrol policy can be delineated by a map π from the state space to the action space $\pi\colon \Omega \to A$. Because the patroller can only move to a node adjacent to the current node, a specific mapping $\mathbf{s} \to j$ is feasible if and only if $a_{l(\mathbf{s}),j} = 1$. We use $\mathcal{A}(\mathbf{s}) = \{j\colon a_{l(\mathbf{s}),j} = 1\}$ to denote the set of feasible actions—or, equivalently, the set of nodes the patroller can move to—when the process is in state $\mathbf{s}$.

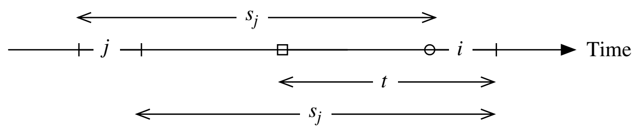
The transition probability of this MDP is deterministic. If the patroller next visits node $i \in \mathcal{A}(\mathbf{s})$ when in state $\mathbf{s}$, the system will transition to state $\tilde{\mathbf{s}} = (\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n)$, where $\tilde{s}_i = 1$, and $\tilde{s}_j = s_j + 1$ for $j \neq i$. For notational simplicity, we define the transition function $\phi(\mathbf{s}, i)$ to specify the resulting state if the patroller visits node i in state $\mathbf{s}$. Namely, $\phi(\mathbf{s}, i) = \tilde{\mathbf{s}}$.

To write the cost function for this MDP, suppose the current state is $\mathbf{s}$, and the patroller visits node i in the next time period. Because the patroller detects attackers at the end of the next time period, the cost incurred in this next time period at node j is equal to the expected number of attackers who complete their attack at node j in that time period, multiplied by c_j. As seen in Figure 1, suppose that at the time marked by a circle, the patroller decides to visit node i next. The attacker arriving to node j at the time marked by a square will complete its attack in the next time period if its attack time X_j falls in $(t-1, t]$. Using the Poisson sampling theorem (see, for example, Proposition 5.3 in Ross 2010), the expected cost incurred at node j is

$$\begin{aligned} C_j(\mathbf{s}, i) &= c_j\lambda_j \int_0^{s_j} P(t-1 < X_j \leqslant t)\,dt \\ &= c_j\lambda_j \int_{s_j-1}^{s_j} P(X_j \leqslant t)\,dt. \end{aligned} \tag{1}$$

The preceding is true for all j, and does not depend on i, because our model assumes the patroller detects the attackers at the end of a time period. Consequently, the cost function for this MDP is $C(\mathbf{s}, i) = \sum_{j=1}^{n} C_j(\mathbf{s}, i)$. Although

Figure 1. This diagram explains the derivation of $C_j(\mathbf{s}, i)$ in (1).



Notes. At the time marked by a circle, the last visit to node j was s_j time units ago and the patroller decides to visit node i next. An attacker arriving to node j at the time marked by the square will complete its attack during the patroller's visit to node i, if its attack time $X_j \in (t-1, t]$. The argument holds for $t \in [0, s_j]$.

$C(\mathbf{s}, i)$ does not directly depend on i, the choice of i affects the state in the next time period, and therefore the cost incurred in the future.

In the case when the attack time X_j is bounded, let

$$B_j \equiv \min\{k\colon k \in \mathbb{Z}^+, P(X_j \leqslant k) = 1\}. \qquad (2)$$

In other words, B_j is the smallest integer that is an upper bound for X_j, $j = 1, \ldots, n$. The cost function in (1) is the same for any $s_j \geqslant B_j + 1$, $j = 1, \ldots, n$. Therefore, for bounded attack times, we can restrict our state space so that $s_j \leqslant B_j + 1$, which allows us to modify our transition function $\tilde{\mathbf{s}} = \phi(\mathbf{s}, i)$ such that $\tilde{s}_i = 1$ and $\tilde{s}_j = \min(s_j + 1, B_j + 1)$ for $j \neq i$. For the remainder of the paper, we will assume that the attack times are bounded, so that the state space is finite.

The objective of this MDP is to minimize the total long-run cost rate among the n nodes. Our state space is finite because the attack time distributions are bounded, and the action space is finite because the number of nodes is finite. Therefore, by Theorem 9.1.8 in Puterman (1994), we only need to consider deterministic, stationary policies.

Because the state transition is deterministic, we can define $\psi_\pi(\mathbf{s}) \equiv \phi(\mathbf{s}, \pi(\mathbf{s}))$ as the resulting state if the patroller applies policy π to state $\mathbf{s}$. For an initial state $\mathbf{s}_0$, policy π will induce an indefinite, deterministic sequence of states, written by $\{\psi_\pi^k(\mathbf{s}_0), k = 0, 1, 2, \ldots\}$, where $\psi_\pi^k = \psi_\pi \circ \psi_\pi^{k-1}$, for $k \geqslant 1$. Because the state space is finite, eventually some state will be visited for a second time, and thereafter the process regenerates itself because the state transition is deterministic under the same policy π. Consequently, after a number of initial transient moves, the sequence $\{\psi_\pi^k(\mathbf{s_0}), k = 0, 1, 2, \ldots\}$ will repeat some cycle indefinitely. Therefore, if we apply policy π to an initial state $\mathbf{s}_0$, we can write the long-run cost rate at node i as

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

which is also equal to the total expected cost incurred in a cycle divided by the cycle length. Furthermore, we call the sequence of nodes corresponding to the cycle a *patrol pattern*.

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

$$C^{\text{OPT}}(\mathbf{s}_0) \equiv \min_{\pi\in\Pi} \sum_{i=1}^{n} V_i(\pi, \mathbf{s}_0), \qquad (3)$$

where Π denotes the class of deterministic, stationary patrol policies. We use the minimum instead of infimum because Π is finite, because the state space is finite. Dividing (3) 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. Whereas $V_i(\pi, \mathbf{s}_0)$ does depend upon $\mathbf{s}_0$, the optimal cost rate $C^{\text{OPT}}(\mathbf{s}_0)$ does not if the graph is connected, because $V_i(\pi, \mathbf{s}_0)$ depends entirely on the patrol pattern generated by $\mathbf{s}_0$ and π. To determine the optimal policy, it is equivalent to find the optimal patrol pattern. If the graph is connected, from any starting state $\mathbf{s}_0$ one can construct a policy π to produce any feasible patrol pattern. Thus, C^{OPT} is the same for all initial states, and we drop its notational dependence on $\mathbf{s}_0$ for the remainder of the paper.

Now that we have defined all the components of the MDP, we can use standard techniques such as linear programming to compute the optimal long-run cost rate. We defer the details to §EC.1.1, which can be found in the electronic companion to this paper. An electronic companion to this paper is available as part of the online version at http://dx.doi.org/10.1287/opre.1120.1149. As discussed in §EC.1.1, this method quickly becomes computationally intractable for problems of moderate size, which motivates the need of efficient heuristics.

3.2. Heuristic Policies on Complete Graphs

To motivate our heuristic policies, we first consider complete graphs. 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. Although the security manager can watch only one video feed at a time, he can switch from any feed to any other feed anytime he wants, which is analogous to a patroller moving from his current node to any node directly. We use indices of the kind developed by Whittle (1988) to develop heuristic policies for the objective function in (3). We refer the reader to Gittins et al. (2011) for a recent account. Whittle index policies for restless bandits have seen near-optimal performance in many other applications (Archibald et al. 2009; Glazebrook et al. 2007, 2009). Below, we outline how to compute a heuristic policy.

To begin, recall that C^{OPT}, defined in Equation (3), denotes the optimal long-run cost rate. First, we relax the problem by extending the class of policies so that the patroller is allowed to visit *multiple nodes* in a time period, as long as the overall long-run visit rate is no greater than 1. To do so, denote by Π^{MN} the set of stationary, deterministic patrol policies

$$\pi\colon \Omega \to \{\boldsymbol{\alpha}\colon \alpha_i \in \{0, 1\} \text{ for } i = 1, \ldots, n\},$$

where $\alpha_i = 1$ if the patroller will visit node i in the next period. Similar to §3.1, the combination of $\pi \in \Pi^{\text{MN}}$ and initial state $\mathbf{s}_0$ induces a patrol pattern. This pattern is more complex than those generated by $\pi \in \Pi$, because now the patroller can visit multiple nodes in one period. Because the same pattern will repeat indefinitely, we can denote by $\mu_i(\pi, \mathbf{s}_0)$ the rate at which the patroller visits node i with policy $\pi \in \Pi^{\text{MN}}$ with initial state $\mathbf{s}_0$, which is just the number of visits to node i in the patrol pattern divided by its length.

We next restrict Π^{MN} to only include policies π that meet the *total-rate* constraint

$$\sum_{i=1}^{n} \mu_i(\pi, \mathbf{s}_0) \leqslant 1, \quad \forall \mathbf{s}_0 \in \Omega. \tag{4}$$

We denote the set of policies that satisfy this constraint as Π^{TR}:

$$\Pi^{\mathrm{TR}} = \left\{ \pi \in \Pi^{\mathrm{MN}} \colon \sum_{i=1}^{n} \mu_i(\pi, \mathbf{s}_0) \leqslant 1, \ \forall \mathbf{s}_0 \in \Omega \right\}.$$

Although both $V_i(\pi, \mathbf{s}_0)$ and $\mu_i(\pi, \mathbf{s}_0)$ depend on the initial state, the optimal long-run cost rate does not, as explained in §3.1. For the remainder of the paper we will write instead $V_i(\pi)$ and $\mu_i(\pi)$ to simplify notations when we can safely ignore their connections to $\mathbf{s}_0$ without ambiguity. The relaxed problem can be formulated by

$$C^{\mathrm{TR}} \equiv \min_{\pi \in \Pi^{\mathrm{TR}}} \sum_{i=1}^{n} V_i(\pi). \tag{5}$$

Comparing Equations (3) and (5), it follows immediately that $C^{\mathrm{OPT}} \geqslant C^{\mathrm{TR}}$ because Π is a subset of Π^{TR}.

Second, we relax the problem again by incorporating the total-rate constraint in (4) into the objective function with a Lagrange multiplier $w \geqslant 0$.

$$\begin{aligned} C(w) &\equiv \min_{\pi \in \Pi^{\mathrm{MN}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + w\left(\sum_{i=1}^{n} \mu_i(\pi) - 1 \right) \right\} \\ &= \min_{\pi \in \Pi^{\mathrm{MN}}} \sum_{i=1}^{n} \{ V_i(\pi) + w\mu_i(\pi) \} - w. \end{aligned} \tag{6}$$

By incorporating a Lagrange multiplier, we can drop the total-rate constraint in (4), so that in (6) the patroller can visit up to n nodes in every time period if he chooses to do so. For any $w \geqslant 0$, we have that

$$\begin{aligned} C^{\mathrm{TR}} = \min_{\pi \in \Pi^{\mathrm{TR}}} \sum_{i=1}^{n} V_i(\pi) &\geqslant \min_{\pi \in \Pi^{\mathrm{TR}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + w\left(\sum_{i=1}^{n} \mu_i(\pi) - 1 \right) \right\} \\ &\geqslant \min_{\pi \in \Pi^{\mathrm{MN}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + w\left(\sum_{i=1}^{n} \mu_i(\pi) - 1 \right) \right\} = C(w). \end{aligned}$$

The first inequality follows because $w \geqslant 0$, and $\sum_{i=1}^{n} \mu_i(\pi) - 1 \leqslant 0$ for any policy $\pi \in \Pi^{\mathrm{TR}}$; the second inequality follows because the total-rate constraint $\sum_{i=1}^{n} \mu_i(\pi) \leqslant 1$ is dropped. Consequently, we have a string of inequalities:

$$C^{\mathrm{OPT}} \geqslant C^{\mathrm{TR}} \geqslant C(w). \tag{7}$$

The optimization problem in (6) breaks up the original problem into n separate problems, each concerning a single node. For instance, node i wants to minimize $V_i(\pi) + w\mu_i(\pi)$, where w can be interpreted as the service charge, when the patroller spends one time period at node i. By solving this problem, it becomes possible to compute an index for each node in each state. An index heuristic policy is for the patroller to visit the node that has the highest index.

3.2.1. Single-Node Problem. This section focuses on the problem facing a single node, when each visit from the patroller costs $w > 0$. Namely, consider the objective function in (6) concerning only node i, and strip off the subscript for simplicity

$$\min_{\pi \in \Pi^{\mathrm{MN}}} V(\pi) + w\mu(\pi). \tag{8}$$

We consider a similar MDP to the one described in §3.1, with the state being the time since the last patrol visit to this node. For the single-node problem a policy $\pi \in \Pi^{\mathrm{MN}}$ simplifies to a binary decision: visit the node or wait. The objective function is to minimize the long-run cost rate, which includes the cost due to not detecting an attack, and the service cost due to a patroller's visit. Because the state space and action space are both finite, we only need to consider deterministic, stationary policies (see Puterman 1994, Theorem 9.1.8.). That is, the optimal action—whether the patroller should visit the node—depends only on the number of periods since the last patrol visit. Because the state increases by 1 each time period without a patrol visit, and returns to 1 after a visit, it is sufficient to consider a policy of this type: Do not visit in states $1, 2, \ldots, k-1$, and visit in state k, where k is a positive integer. In other words, we only need to consider those policies that visit the node once every k time periods, for $k = 1, 2, \ldots$.

We next write out the objective function in (8) when the patroller visits the node once every k time periods. We say a renewal occurs each time the patroller visits the node, so the cycle time (time between renewals) is k. An attacker arriving at time t following a renewal, $0 \leqslant t < k$, will complete its attack if its attack time is no greater $k - t$. Using a Poisson sampling result (for example, Proposition 5.3 in Ross 2010), the number of successful attacks in a cycle follows a Poisson distribution with expected value equal to

$$\lambda \int_0^k P(X \leqslant k - t)\, dt = \lambda \int_0^k P(X \leqslant t)\, dt.$$

Because each successful attack costs c, and a patrol visit costs w, the long-run cost rate is

$$f(k) \equiv \frac{c\lambda \int_0^k P(X \leqslant t)\, dt + w}{k}. \tag{9}$$

for $k = 1, 2, \ldots$. Thus, solving (8) is equivalent to finding k to minimize $f(k)$ in (9).

To minimize $f(k)$, we first compute

$$\begin{aligned} &f(k+1) - f(k) \\ &= \frac{1}{k(k+1)} \left(c\lambda k \int_0^{k+1} P(X \leqslant t)\, dt \right. \\ &\qquad \left. - c\lambda(k+1) \int_0^k P(X \leqslant t)\, dt - w \right) \\ &= \frac{1}{k(k+1)} \left(c\lambda k \int_k^{k+1} P(X \leqslant t)\, dt - c\lambda \int_0^k P(X \leqslant t)\, dt - w \right). \end{aligned}$$

By setting $f(k+1)=f(k)$, we can find the per-visit cost w that makes the patroller indifferent between visiting the node once every k time periods, or once every $k+1$ time periods. The solution will help us characterize the optimal policy minimizing $f(k)$, and is defined by

$$W(k)\equiv c\lambda\left(k\int_k^{k+1}P(X\leqslant t)\,dt-\int_0^k P(X\leqslant t)\,dt\right) \tag{10}$$

for $k=1,2,\ldots.$ Because X is bounded by a constant B, for $k\geqslant B$, we have that

$$\begin{aligned}W(k)&=c\lambda\left(k-\int_0^k P(X\leqslant t)\,dt\right)=c\lambda\int_0^k P(X>t)\,dt\\&=c\lambda\int_0^B P(X>t)\,dt=c\lambda E[X].\end{aligned} \tag{11}$$

In addition, for $k=0$, Equation (10) implies $W(0)=0$. The next theorem uses the functions $W(k)$, $k\geqslant 0$, to characterize the optimal policy minimizing the objective in (8).

Theorem 1. *The function $W(k)$ defined in* (10) *is nondecreasing in k. In addition, for the single-node problem defined in* (8), *if $w\in[W(k-1),W(k)]$, then it is optimal to visit the node once every k time periods, for $k=1,2,\ldots.$ Moreover, if $w\geqslant c\lambda E[X]$, it is optimal not to visit the node at all.*

The proof of this theorem is deferred to §EC.2.1. From the theorem, we can interpret $W(k)$ as the maximum per-visit cost for the policy that visits the node in state k (once every k time periods) to be optimal.

3.2.2. Index Heuristic (IH). To develop a heuristic based on indices, affix a subscript i in Equation (10) to define

$$W_i(k)\equiv c_i\lambda_i\left(k\int_k^{k+1}P(X_i\leqslant t)\,dt-\int_0^k P(X_i\leqslant t)\,dt\right) \tag{12}$$

as the index of node i if the last patrol visit to node i took place k time periods ago, or equivalently, if node i is in state k.

To implement a heuristic based on these indices, we choose the initial state by supposing that the patrol area has been neglected for a long time. Therefore, initially we set $s_i=B_i+1$. In the first time period, the patroller simply begins his patrol at a node that has the highest index value. For each subsequent time period, the patroller compares the indices of all nodes (including the current node) and visits the one that has the highest index value. We call the preceding patrol policy the *index heuristic* (IH). Mathematically, whenever in state $\mathbf{s}=(s_1,s_2,\ldots,s_n)$, the patroller next visits node j if $W_j(s_j)=\max_{i=1,2,\ldots,n}W_i(s_i)$. In case there is a tie, break the tie arbitrarily.

Recall from Theorem 1 that $W_i(k)$ is nondecreasing in k for all i. Therefore, a node's index value increases with each time period without a patrol visit and returns to its smallest possible value immediately after a patrol visit. Because $\lambda_i=p_i\Lambda$ for all i and the indices are used for comparison across nodes, an equivalent index is to replace λ_i with p_i in (12).

3.2.3. Lower Bound. Recall from (7) that $C^{\text{OPT}}\geqslant C^{\text{TR}}\geqslant C(w)$, where $C(w)$ represents the optimal long-run cost rate when each node operates independently with a per-visit cost w. The value $C(w)$ in (6) is a lower bound for the optimal cost rate C^{OPT} for any $w\geqslant 0$. In this section, we compute the tightest such lower bound, namely, $C^{\text{TR}}=\max_{w\geqslant 0}C(w)$.

Recall that $W_i(k)$ represents the per-visit cost that makes node i indifferent between receiving a patrol visit once every k time periods, or once every $k+1$ time periods. For a given per-visit cost w, we can define

$$K_i(w)\equiv\begin{cases}\infty, & \text{if } w\geqslant W_i(B_i);\\ \min\{k\colon W_i(k)>w\}, & \text{otherwise.}\end{cases}$$

From Theorem 1, $K_i(w)$ represents the optimal interval between visits at node i when each patrol visit costs w. According to this definition, when there are multiple optimal intervals, we break ties by choosing the longest such interval. Consequently, we can rewrite $C(w)=\sum_{i=1}^n C_i(w)-w$, where

$$\begin{aligned}C_i(w)&\equiv f_i(K_i(w))\\&=\begin{cases}c_i\lambda_i, & \text{if } w\geqslant W_i(B_i);\\ \dfrac{c_i\lambda_i\int_0^{K_i(w)}P(X\leqslant t)\,dt+w}{K_i(w)}, & \text{otherwise.}\end{cases}\end{aligned}$$

The second part of the preceding is derived by affixing a subscript i in (9).

The function $C_i(w)$ represents the optimal long-run cost rate for node i if the patroller charges w for each patrol visit. First, $C_i(w)$ must be nondecreasing in w, because the node can always do better with a smaller service charge by using the same service interval. Second, $C_i(w)$ is piecewise linear, with turning points occurring only at $w=W_i(k)$, for $k=1,2,\ldots,B_i$. Third, $C_i(w)$ is concave, because for $w\neq W_i(k)$, the function $K_i(w)$ remains a constant and $C_i'(w)=1/K_i(w)$, which is nonincreasing in w, because $K_i(w)$ is nondecreasing in w. Consequently, $C(w)=\sum_{i=1}^n C_i(w)-w$ is also piecewise linear and concave. Therefore, it is straightforward to compute $\max_w C(w)$.

The optimal solution that maximizes $C(w)$ can either be a point or a line segment. When $w\neq W_i(k)$ for some i, k, we have that $C'(w)=\sum_{i=1}^n 1/K_i(w)-1$. That is, $C'(w)$ is a step function that changes value at $W_i(k)$ for some i, k, and is nonincreasing. In the case the optimal solution is unique, denoted by w^*, we need to find w^* such that

$$w<w^*\iff\sum_{i=1}^n\frac{1}{K_i(w)}>1,$$

$$w>w^*\iff\sum_{i=1}^n\frac{1}{K_i(w)}<1.$$

In the case where the optimal solutions consist of a line segment, we need to find w^* such that $\sum_{i=1}^n 1/K_i(w^*)=1$.

3.2.4. Optimality of the Index Heuristic in Special Cases. The index machinery has been used to develop heuristics in the context of restless bandits (Gittins et al. 2011, Whittle 1988). In some special cases, it is possible to show that the index heuristics of this kind produce the optimal solution. To intuitively understand how this might occur, observe that there must exist w^* such that $C(w^*) = \max_{w\geqslant 0} C(w)$, and $w^* = W_i(k)$ for some i, k (see §3.2.3). For ease of explanation, suppose that w^* and the node are both unique and that the latter is labeled 1. The Lagrangian relaxation with $w = w^*$ will be solved by any (possibly inadmissible) patrol pattern π_1 that visits each node i every $K_i(w^*)$ time units, and by a second pattern π_2 that modifies π_1 by visiting node 1 every $K_1(w^*) - 1$ time units. Some randomization between these two patterns, denoted $\alpha \otimes \pi_1 + (1-\alpha) \otimes \pi_2$, will achieve $\max_{w\geqslant 0} C(w)$ and also have the property that the overall rate of patrol visits is 1. If one can show that from some finite time on, the IH produces the same pattern of costs to each node as does $\alpha \otimes \pi_1 + (1-\alpha) \otimes \pi_2$, then the IH must be optimal. It turns out that this is always the case when $n = 2$, which we state in Theorem 2. For larger problems, this matching of cost rates between the IH and $\alpha \otimes \pi_1 + (1-\alpha) \otimes \pi_2$ can be achieved—when there are Q patrollers, Q a suitably chosen integer—by creating Q shifted copies of patrol patterns that are inadmissible for a single patroller. Theorem 3 formalizes this idea.

Theorem 2. *If $n = 2$, then the IH is optimal and $C^{\text{IH}} = C^{\text{OPT}} = \max_{w\geqslant 0} C(w)$.*

The proof of this theorem is deferred to §EC.2.2. From Theorem 2, we see that IH is optimal for very small graphs ($n = 2$). Unfortunately, the theorem does not hold for $n \geqslant 3$, as we have seen in counterexamples. In order to explore its performance for very large graphs ($n \to \infty$), it is natural to consider the asymptotic regime discussed by Weber and Weiss (1990, 1991). In this regime, the number of nodes of the graph and the number of patrollers go to infinity in fixed proportion. We are able to establish Theorem 3 and Corollary 1, which make considerably stronger claims than are generally available for restless bandits, and which are not subject to the associated sufficient conditions that are difficult to verify. Please note that this is the only place we consider multiple patrollers in this paper.

Recall that in our formulation, each node i is characterized by the triple (c_i, λ_i, F_i), where F_i is the distribution function of the attack time $X_i, i = 1, \ldots, n$. We now consider a Q-fold amplification of our base 1-patroller-n-node problem. The Q-fold amplification consists of Q autonomous patrollers operating among nQ nodes, the latter consisting of Q nodes with the characteristics (c_i, λ_i, F_i) for each i, $i = 1, \ldots, n$. The new graph with nQ nodes is still complete, namely, that every node is accessible from every other node in a single time step. It will assist to label the nodes (i, j), for $i = 1, \ldots, n$ and $j = 1, \ldots, Q$, where nodes (i, j), $j = 1, \ldots, Q$, share the characteristics (c_i, λ_i, F_i). We denote objects related to the Q-fold amplification via a superscript Q, and hence write $C^{\text{OPT}, Q}$, $C^Q(w)$, etc. The Lagrangian relaxation appropriate for the Q-fold amplification replaces (4) by the constraint $\sum_{i=1}^n \sum_{j=1}^Q \mu_{ij}(\pi) \leqslant Q$. It is evident that $C^Q(w) = Q \cdot C(w)$, where $C(w)$ is, as usual, defined with respect to the base 1-patroller-n-node problem. Hence, there is a common maximizer w^* for both $C^Q(\cdot)$ and $C(\cdot)$. We shall use the notations $W_i(k)$, $K_i(w)$ unambiguously in what follows without the need for the second node identifier. We are now able to state that for any base problem, certain Q-fold amplifications are such that the corresponding Lagrangian relaxation is tight and that there is an optimal policy, which from some time on always visits Q nodes of highest index.

Theorem 3. *For complete graphs, there exists $Q \in \mathbb{Z}^+$, such that*

$$C^{\text{OPT}, NQ} = \max_{w\geqslant 0} C^{NQ}(w), \quad \forall N \in \mathbb{Z}^+.$$

Moreover, for any such NQ-amplification there exists an optimal policy, which from time Q onward always visits NQ nodes of highest index.

The proof of this theorem is deferred to §EC.2.3. The following result concerns asymptotic tightness of the Lagrangian relaxation and asymptotic optimality of index policies for problems with complete graphs. It is a simple consequence of Theorem 3 and its proof.

Corollary 1. *For complete graphs the Lagrangian relaxation in* (6) *is asymptotically tight in the sense that for any base problem, $\lim_{m\to\infty} C^{\text{OPT},m}/m = \max_{w\geqslant 0} C(w)$. Moreover, there exists $Q \in \mathbb{Z}^+$ and a sequence of policies $\{\pi_m, m \in \mathbb{Z}^+\}$ such that π_m is a policy for the m-amplification choosing m nodes of highest index at all times from time Q onward and satisfying $\lim_{m\to\infty} C^{\pi_m, m}/m = \max_{w\geqslant 0} C(w)$.*

In this subsection, we showed the optimality of the IH, in certain special cases, by proving that it achieves the lower bound based on the Lagrangian relaxation. There are other cases where we can construct an optimal policy from the Lagrangian lower bound. For instance, if the Lagrangian relaxation results in $K_i(w^*) = 1/n$, $i = 1, \ldots, n$, for a complete graph or a circle graph with n nodes, then the optimal policy is simply any Hamiltonian cycle. That said, it is extremely unlikely that tractable approaches to the development of patrol patterns achieving $C(w^*)$ can be developed in general. For example, even in the special case when $K_i(w^*) = 1/n$ for $i = 1, \ldots, n$, determining whether there exists a Hamiltonian cycle in an arbitrary graph is NP-complete (Karp 1972). Intuitively speaking, the IH uses a greedy method to find a feasible patrol pattern that comes close to the optimal policy for the relaxed Lagrangian problem. The next section extends the ideas of the IH to develop heuristic policies on arbitrary graphs.

3.3. Heuristic Policies on Arbitrary Graphs

On complete graphs, the patroller may visit any node at any time, whereas on an arbitrary graph, the patroller can only visit a node that is adjacent to his current node. If the patroller just visits the node that has the highest index value among all adjacent nodes, the patroller may easily become stuck in a subgraph, especially on a 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. There are two natural ways to formulate the aggregate index. For comparison purposes in our numerical study, we also include a myopic heuristic based only on the expected cost that can be avoided for the next time period.

3.3.1. Index Reward Heuristic (IRH). First, we can interpret the index of the *selected* node as a reward. With this interpretation, an l-step look-ahead aggregate index of a path is the sum of the indices 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 visit based on the largest aggregate index among all those paths. Even though this heuristic computes the aggregate index for an l-step path, the aggregate index is used to determine only the next node. Once at the next node, a new l-step look-ahead aggregate index is computed. We call this heuristic the *index reward heuristic* (IRH).

Regardless of the choice of l, the IRH is a function that maps from a state to a node. Because the state transition is deterministic, whenever the process enters the same state, the IRH will generate the same patrol sequence. Therefore, the patrol schedule generated by the IRH 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.

One interesting and important question is, does the performance of the heuristic always improve when l increases? The answer is no. When comparing $l = 1$ and $l = 2$, the IRH may return the same patrol pattern, or may return two distinct patterns. When the two patterns are distinct, the pattern generated with $l = 2$ may perform better than the pattern generated with $l = 1$, or it may perform worse. The time it takes to compute a patrol pattern in a complete graph is proportional to n^l, because we need to compare all n^l paths of length l to determine the next node. For these reasons, it does not make sense to compute a patrol pattern by setting $l = 2$ without first examining $l = 1$.

We call it the *index reward heuristic with depth d*, or IRH(d), if we compare the d patrol patterns generated by look-ahead windows $l = 1, 2, \ldots, d$, and choose the best one. Consequently, by definition, the IRH(d) improves (weakly) as d increases. For complete graphs, IH and IRH(1) are the same.

3.3.2. Index Penalty Heuristic (IPH). Second, 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 visit 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. By definition, the IPH(d) improves (weakly) as d increases. Note that IPH(1) and IRH(1) are the same.

3.3.3. Myopic Heuristic (MH). We also consider a myopic heuristic in our numerical study. In state $\mathbf{s}$, if the patroller visits node i next, then the expected number of attacks he can detect is $\lambda_i \int_0^{s_i} P(X_i > t)\,dt$, which follows from the Poisson sampling theorem (see, for example, Proposition 5.3 in Ross 2010). The expected cost that can be avoided—or reward gained—if the patroller visits node i in state $\mathbf{s}$ is therefore

$$R(\mathbf{s}, i) = c_i \lambda_i \int_0^{s_i} P(X_i > t)\,dt. \tag{13}$$

A myopic heuristic policy that looks ahead l time periods compares all feasible paths of length l and chooses the next node to visit according to the path that gives the highest total reward gained over that path. We call it the *myopic heuristic with depth d*, or MH(d), if we compare the d patrol patterns generated by look-ahead windows $l = 1, 2, \ldots, d$, and choose the best one. By definition, the MH(d) improves (weakly) as d increases.

3.3.4. Lower Bound. To derive a lower bound for the optimal value on an arbitrary graph, note that the patroller can do no worse if all the nodes were connected. Therefore, the lower bound derived in §3.2.3 is also a lower bound for arbitrary graphs. However, because that lower bound does not take into account graph structure, it can be quite loose in general, especially for sparse graphs such as trees.

We next present a linear program that computes a tighter lower bound by taking into account graph structure. To begin, consider a single patrol pattern, and denote by y_{ik} the rate at which the patroller enters node i exactly k time units after his previous visit to node i. For instance, if the patrol pattern is 1-1-2-1-3-1-3-2, then $y_{1,2} = 2/8$, whereas $y_{1,1} = y_{1,3} = y_{2,3} = y_{2,5} = y_{3,2} = y_{3,6} = 1/8$. From the definition, it follows clearly that if node i is present in the patrol pattern, then

$$\sum_{k=1}^{\infty} k y_{ik} = 1. \tag{14}$$

If node i is not in the patrol pattern, then $y_{ik} = 0$ for all k, so $\sum_{k=1}^{\infty} k y_{ik} = 0$.

In addition, let x_{ij} denote the rate at which the patroller moves from node i to node j. With the same patrol pattern

1-1-2-1-3-1-3-2, we have that $x_{1,1} = x_{1,2} = x_{3,1} = x_{3,2} = 1/8$, and $x_{1,3} = x_{2,1} = 2/8$. By definition,

$$\sum_{j=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ji}, \tag{15}$$

which is also the rate at which the patroller visits node i, $i = 1, 2, \ldots, n$. Condition (15) ensures flow balance at each node. The variables x_{ij} and y_{ik} are connected, with two obvious equations being

$$y_{i,1} = x_{ii}, \quad i = 1, 2, \ldots, n, \tag{16}$$

$$\sum_{k=1}^{\infty} y_{ik} = \sum_{j=1}^{n} x_{ji}, \quad i = 1, 2, \ldots, n. \tag{17}$$

Constraint (16) follows because either side represents the rate at which the patroller visits node i in two consecutive time periods. Constraint (17) follows because either side represents the long-run rate at which the patroller enters node i.

Now, suppose we allow a randomized policy, such that the patroller uses a set of patrol patterns and selects each pattern with a predetermined probability. For a randomized policy, we define y_{ik} and x_{ij} as the *weighted average* over their counterparts in individual patrol patterns. We can interpret y_{ik} as the *expected rate* at which the patroller enters node i exactly k time units after his previous visit to node i, and x_{ij} the *expected rate* the patroller moves from node i to node j. Whereas constraints (15), (16), and (17) still hold for randomized policies, (14) holds if and only if node i is present in all patrol patterns in the random mix. Hence, for an arbitrary randomized policy, the constraint in (14) becomes an inequality:

$$\sum_{k=1}^{\infty} k y_{ik} \leqslant 1, \quad i = 1, 2, \ldots, n. \tag{18}$$

To set up a linear program, however, we cannot deal with infinitely many terms y_{ik}, $k = 1, 2, \ldots$. Recall from (2) that B_i denotes the smallest integer such that $P(X_i \leqslant B_i) = 1$, $i = 1, \ldots, n$. Hereafter we redefine y_{i,B_i} to denote the expected rate at which the patroller enters node i with the previous visit *at least* B_i time units ago. Constraints in (17) and (18) can be rewritten as

$$\sum_{k=1}^{B_i} y_{ik} = \sum_{j=1}^{n} x_{ji}, \quad i = 1, 2, \ldots, n, \tag{19}$$

$$\sum_{k=1}^{B_i} k y_{ik} \leqslant 1, \quad i = 1, 2, \ldots, n. \tag{20}$$

To formulate the objective function, recall from (13) that if the patroller visits node i exactly k time units after his previous visit to node i, then the expected cost that can be avoided is $R_i(k) \equiv c_i \lambda_i \int_0^k P(X_i > t)\,dt$. Because $R_i(k) = c_i \lambda_i E[X_i]$, if $k \geqslant B_i$, the long-run cost rate at node i is

$$c_i \lambda_i - \sum_{k=1}^{B_i} y_{ik} R_i(k). \tag{21}$$

Finally, the linear program is

$$\min_{x_{ij},\, y_{ik}} \sum_{i=1}^{n} \left(c_i \lambda_i - \left(\sum_{k=1}^{B_i} y_{ik} R_i(k) \right) \right)$$

$$\text{subject to} \quad x_{ij} = 0, \quad \text{if } a_{ij} = 0, \tag{22}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} = 1, \tag{23}$$

$$x_{ij},\ y_{ik} \geqslant 0, \tag{24}$$

and constraints in (15), (16), (19), and (20).

Constraint (22) observes the edge constraint, and constraint (23) ensures the total rate to be 1. Because each feasible patrol pattern (or a randomization over a set of patrol patterns) yields a feasible solution to this linear program, but not vice versa, the optimal solution to this linear program provides a lower bound for the optimal long-run cost rate.

It is possible to tighten the lower bound further by adding constraints on y_{ik} or constraints that take advantage of specific graph structures. Some of these ideas are presented in §EC.3. In the next section, we use the lower bounds produced by this linear program to prepare Tables 2 and 3.

3.4. Numerical Experiments

We consider five graph types in our numerical experiments.

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 only one video feed at a time, but can switch from any feed to any other feed directly, which is analogous to a patroller moving to any node directly.
2. Line graph: A line graph is applicable to an airborne 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 (Dunnigan 1992) 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.

Although our heuristic works 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 the patroller will never detect anything. Because 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. For each node, the attack time is equally likely to follow these three distributions. In the case of a deterministic attack time, we generate a uniform random variable $U[1, n]$ 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.

In order to better interpret the results, we set $c_i = 1$ for all i, so that $\sum_{i=1}^{n} V_i(\pi)/\Lambda$ is the long-run proportion of attackers that evade detection, or equivalently, the limiting probability an attacker will evade detection. To generate p_i (probability of attacking node i), we first generate uniform variables $u_i \sim U[0, 1]$, and then normalize them so that $p_i = u_i / \sum_{j=1}^{n} u_j$. Recall that the value of Λ is inconsequential.

To obtain a patrol pattern, recall from §3.2.2 that we set the initial state by assuming that the entire graph has not been patrolled for a long time. We then use a heuristic to generate a sequence of patrol nodes and determine the corresponding patrol pattern when a state repeats itself. If no patrol pattern emerges after 2,000 time periods, we then use the entire patrol schedule of length 2,000 as a proxy for the actual patrol pattern, which happens occasionally for some of the larger graphs analyzed in this numerical study.

Our first experiment is to compare the three heuristics in §3.3, namely the IRH, the IPH, and the MH. The IH for complete graphs is the same as IRH(1) and IPH(1), although it does not show up explicitly in our numerical study. We examine graphs of size $n = 6$, for which we can compute the optimal solution. To make the point and save space, we present results for only two graph types—complete graph and line graph—because these are the two extremes in terms of graph connectivity. For the same number of nodes, the complete graph represents the easiest graph structure for the patroller, whereas the line graph represents the most difficult one.

For each graph type we generate 1,000 scenarios. A scenario consists of the adjacency matrix for the graph, the p_i values, and the attack time distributions. The p_i values and attack time distributions are the same for corresponding scenarios across different graph types. For each scenario, we compute the optimal solution (evasion probability) and report the heuristics in terms of their percentage excess over the optimal evasion probability. Over the 1,000 scenarios, we report the mean, the 50th, 75th, and 90th percentile in Table 1. Recall that IRH(1) and IPH(1) are identical, but as d increases, IRH(d) improves marginally, whereas IPH(d) improves quite significantly. Also seen in Table 1, the IPH outperforms the MH by a sizable margin.

One would not expect the MH to perform particularly well, as is the case in most dynamic resource allocation problems. However, it may come as somewhat of a surprise that the IPH outperforms the IRH when $d > 1$. To intuitively understand the downside of the IRH, recall that $W_i(k)$ increases in k for node i. When we look ahead for l time periods and treat the index value as a reward, then at times the IRH will wait on node i in order for its index to increase, so as to collect a higher reward later on. This strategy may backfire if the patroller ends up wasting time going to other nodes that benefit little from a patrol visit. We provide an example with two nodes to illustrate this point.

Table 1. Performance of the three heuristics on complete graphs and line graphs with six nodes, reported as the percentage excess over the optimal evasion probability.

		Complete graph				Line graph			
Heuristic	Depth (d)	Mean	50th	75th	90th	Mean	50th	75th	90th
IRH	1	2.72	0.00	2.34	7.64	12.16	3.96	14.94	33.84
IRH	2	2.31	0.00	1.85	6.33	4.95	0.28	6.58	13.66
IRH	3	2.28	0.00	1.79	6.29	4.16	0.00	4.92	12.15
IPH	1	2.72	0.00	2.34	7.64	12.16	3.96	14.94	33.84
IPH	2	0.76	0.00	0.41	2.16	1.39	0.00	0.41	4.04
IPH	3	0.57	0.00	0.32	1.85	0.50	0.00	0.00	1.04
MH	1	17.28	11.78	22.74	39.81	15.28	8.25	21.88	40.44
MH	2	4.56	0.77	5.54	13.02	6.30	2.31	8.90	17.71
MH	3	1.54	0.00	1.55	5.03	2.83	0.00	3.16	8.76

Note. We randomly generate 1,000 scenarios and report the mean, the 50th percentile, the 75th percentile, and the 90th percentile.

Example. Consider an example with $n = 2$ and $\Lambda = 1$. Let $p_1 = 0.1$, $p_2 = 0.9$, and $c_1 = c_2 = 1$. In addition, the attack time distributions are deterministic at both nodes, with $P(X_1 = 2) = P(X_2 = 2.5) = 1$. The optimal policy is clearly for the patroller to alternate between the two nodes (patrol pattern 1-2), because it detects all attacks, therefore yielding a cost rate of 0.

To see how the IRH performs, first write the indices in a matrix form as follows:

$$\begin{pmatrix} W_1(1) & W_1(2) & W_1(3) \\ W_2(1) & W_2(2) & W_2(3) \end{pmatrix} = \begin{pmatrix} 0 & 0.2 & 0.2 \\ 0 & 0.9 & 2.25 \end{pmatrix}.$$

Because both attack times are bounded by 3, $W_i(k) = W_i(3)$, for $i = 1, 2$ and $k > 3$. When the look-ahead window is 1, the IRH will pick node 1 in state $(2, 1)$, and pick node 2 in state $(1, 2)$, so the resulting patrol pattern is 1-2, the optimal one. When the look-ahead window is 2, however, in state $(1, 2)$ the IRH will pick node 1, because by looking ahead for 2 time periods, the IRH can collect a much higher reward of 2.25 if it waits for another time period before visiting node 2. The resulting patrol pattern is 1-1-2, yielding a cost rate of 0.15. □

From this point on, we will focus on the IPH. As discussed earlier, the performance of IPH(d) improves as d increases, but the computation takes more time. As shown in Table 1, the improvement is more significant in the line graph than in the complete graph. We next test IPH(d) on five graph types with $d = 1, \ldots, 5$ to study how the performance improves at the cost of computation time, and plot the results in Figure 2. Each line corresponds to a graph type, with five points corresponding to $d = 1$ on the left to $d = 5$ on the right. As d increases, the performance gets better (moving lower) but computation takes longer (moving to the right).

As seen in Figure 2, when d increases, the improvement on the performance is more pronounced on graphs with fewer edges (line, circle, and random trees), and less so on graphs that are well connected (complete and hexagon grid). Intuitively, with a small look-ahead window, the patroller is more likely to get stuck in a subgraph of a sparse graph than in a subgraph of a well-connected graph. Therefore, it is reasonable to set d based on the graph structure, with a small value when the graph is well connected (such as a complete graph) and a large value when it is not (such as a line graph). We propose a Modified Index Penalty Heuristic (MIPH), where we set the depth to

$$d = 1 + \lceil \text{average distance between all pairs of nodes} \rceil.$$

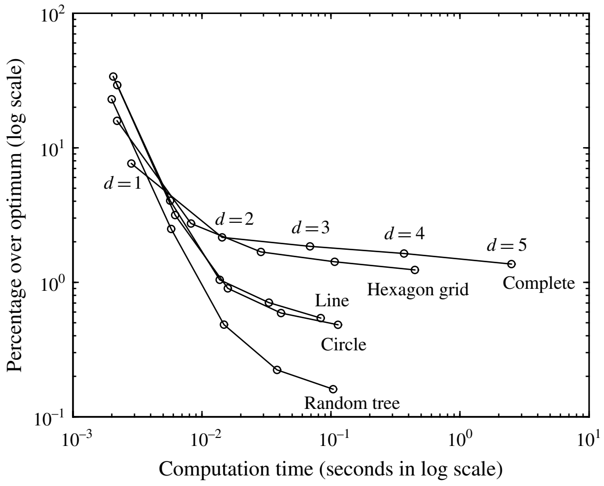
In other words, the depth is set to 2 for a complete graph, and increases as the graph becomes less connected.

Table 2 reports the performance of the MIPH for 5 graph types by comparing them to the optimal solution. Because the state space in the MDP depends on the graph types, for complete graphs we can compute the optimal solution for up to 7 nodes, while for line graphs we can do so for up to 14 nodes. In all cases, we randomly generate 1,000 scenarios and report the mean, the 50th, 75th, and the 90th percentiles. As seen in Table 2, the MIPH performs uniformly well across all graph types. In particular, there is little evidence of any degradation in performance as the graph size n grows within our range of interest. Table 2 also gives the average depth and the average computation time. Overall, we find the MIPH offers a good balance between performance and computation time. The last column in Table 2 reports how the lower bound discussed in §3.3.4 compares with the optimal solution on average. For each scenario, we compute $(\text{LB} - \text{Opt})/\text{Opt}$ in percentage, and report the average over 1,000 scenarios. For all graph types, the lower bounds are about 1% below the optimal solution for $n = 6, 7$, but they gradually degrade as n increases.

To assess how much our heuristic improves over a naïve patrol strategy, we consider two graph types: line graph and circle graph. For a line graph, a naïve patrol moves back and forth between two end nodes, spending just one time period at each end node. For a circle graph, a naïve patrol circles around all nodes. For $n = 6$, among the same 1,000 scenarios reported in Table 2, on average the naïve patrol produces an evasion probability 20.57% over optimum for line graphs, and 19.57% over optimum for circle graphs. In either case, on average our heuristic produces an evasion probability less than 0.5% over optimum.

To conclude our numerical study in this section, we next look at larger problems by comparing the MIPH with the lower bound discussed in §3.3.4. We only examine complete graphs, because the quality of the available lower

Figure 2. This figure displays, for the random-attacker case, the IPH performance against computation time for different d, on five graph types with $n = 6$.



Notes. The performance is the 90th percentile over 1,000 random scenarios, reported as percentage over optimum. Each line corresponds to one graph type, with $d = 1, 2, 3, 4, 5$ from left to right.

Table 2. Performance of the MIPH in the random-attacker case, reported as percentage excess over the optimal evasion probability; the last column reports the mean of (LB − Opt)/Opt in percentage, where the lower bound is computed using the linear program in §3.3.4.

Graph	# nodes	Mean	50th	75th	90th	Avg depth	Avg time (10^{-2} sec)	Lower bound
Complete	6	0.76	0.00	0.41	2.16	2.0	1.3	−1.12
Complete	7	1.11	0.00	1.11	3.12	2.0	2.3	−1.26
Line	6	0.39	0.00	0.00	0.71	4.0	1.9	−0.67
Line	7	0.33	0.00	0.00	0.60	4.0	2.5	−1.30
Line	8	0.51	0.00	0.00	1.21	4.0	3.3	−1.80
Line	9	0.42	0.00	0.00	1.27	5.0	11.0	−2.89
Line	10	0.48	0.00	0.14	1.38	5.0	13.6	−3.50
Line	11	0.44	0.00	0.26	1.47	5.0	16.7	−4.33
Line	12	0.34	0.00	0.03	1.08	6.0	56.2	−5.02
Line	13	0.41	0.00	0.25	1.34	6.0	86.9	−6.03
Line	14	0.53	0.00	0.53	1.59	6.0	100.7	−6.39
Circle	6	0.32	0.00	0.00	0.90	3.0	1.1	−0.44
Circle	7	0.43	0.00	0.00	1.50	3.0	1.4	−1.03
Circle	8	0.36	0.00	0.00	1.26	4.0	4.7	−1.39
Circle	9	0.41	0.00	0.14	1.42	4.0	5.9	−2.12
Random tree	6	0.26	0.00	0.00	0.27	3.7	3.5	−0.59
Random tree	7	0.25	0.00	0.00	0.32	3.9	5.5	−0.98
Random tree	8	0.25	0.00	0.00	0.84	4.0	8.3	−1.53
Random tree	9	0.27	0.00	0.00	0.71	4.0	11.3	−2.07
Hexagon grid	6	0.46	0.00	0.00	1.68	3.0	2.3	−0.68
Hexagon grid	7	0.52	0.00	0.22	1.68	3.0	3.7	−1.14
Hexagon grid	8	0.61	0.00	0.60	2.03	3.0	5.1	−1.76

Table 3. Performance of the MIPH in the random-attacker case on complete graphs, reported as the percentage excess over the lower bound, which is computed using the linear program in §3.3.4.

# nodes	Mean	50th	75th	90th
6	1.96	0.75	2.50	5.09
9	2.51	1.81	3.27	5.23
12	2.28	1.67	3.05	4.78
15	2.13	1.75	2.76	3.98
18	2.07	1.76	2.58	3.80

bound for the other graph types degrades as n increases (see Table 2), so such a comparison does not provide useful information. As shown in Table 3, on average the MIPH exceeds the lower bound by about 2% (both mean and 50th percentile), and the 90th percentile is about 5% over the lower bound. These numbers are very encouraging, because they suggest that the MIPH produce excellent results for complete graphs up to 18 nodes.

4. Patrol Against Strategic Attackers

This section concerns the situation when the attacker actively chooses which node to attack, in order to maximize the cost incurred due to its attack. In other words, the attacker and the patroller play a simultaneous-move two-person zero-sum game, with the patroller trying to minimize the cost, and the attacker trying to maximize it. The patroller decides how to patrol the graph, whereas the attacker chooses which node to attack.

One way to formulate this problem is to use the model framework in §3. For a given patrol policy, the ratio $V_i(\pi)/\lambda_i$ represents the long-run average cost incurred due to an attack at node i. Consequently, the patroller's objective function in this two-person zero-sum game is

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

where Π^R represents the set of randomized policies—all policies that map from the state space Ω to the action space A according to a probability distribution. Because $V_i(\pi)$ scales proportionally with λ_i, the ratio $V_i(\pi)/\lambda_i$ does not depend on λ_i.

4.1. Optimal Policy

By modifying the linear program that computes the optimal solution for the random-attacker case, it is possible to compute the optimal value in (25). We defer the details to §EC.1.2. This method produces the optimal solution used in our numerical studies, but becomes computationally intractable for moderate-size problems.

4.2. Heuristic Policies

In a two-person zero-sum game, it is often the case that the optimal strategy for either player is a mixed strategy. A mixed strategy for the attacker is a probability distribution over the nodes to attack. A mixed strategy for the patroller is a distribution that defines, for each state of the system, the probability that the patroller will move to each adjacent node.This interpretation allows the linear program formulation described in §EC.1.2 to solve the problem optimally. However, this approach quickly becomes computationally intractable as the size of the problem grows.

Another way to randomize a patroller's strategy is to begin with a set of feasible patrol patterns, and let the patroller choose each pattern *in that set* with a certain probability and repeat that pattern indefinitely. If the set includes all feasible patterns (there are infinitely many), then the resulting mixed strategy over patterns would achieve optimality.

Because we cannot examine an infinite number of patrol patterns, we propose a heuristic to compute a mixed strategy from a finite set of selected patrol patterns. Given a finite set of patrol patterns, denoted by $\mathcal{S} = \{\xi_1, \xi_2, \ldots, \xi_m\}$, with $|\mathcal{S}| = m$, we can formulate a *different* two-person zero-sum game between the attacker and the patroller in the standard matrix form. In this game matrix, row i corresponds to the attacker choosing node i to attack, and column j corresponds to the patroller choosing patrol pattern ξ_j, with $i = 1, \ldots, n$ and $j = 1, \ldots, m$. It is then straightforward to set up a linear program to solve this two-person zero-sum game (see §3.10 in Washburn 2003).

Of course, the solution to this $n \times m$ matrix game will not necessarily be the globally optimal mixed strategy, because we only consider a finite set $\mathcal{S}$. The key to the success of this approach is to generate the set $\mathcal{S}$ so that the optimal mixed strategies using only the patrol patterns in $\mathcal{S}$ is close to the global optimum. The major advantage is that the linear program that solves the $n \times m$ matrix game is much smaller than the linear program discussed in §EC.1.2.

To obtain patrol patterns that constitute $\mathcal{S}$, recall that we can use the IPH(d) in §3.2.2 to find a patrol pattern against a given attack probability distribution over n nodes $\mathbf{p} = (p_1, \ldots, p_n)$. Below we discuss how to use the IPH(d) to generate patrol patterns that compose $\mathcal{S}$. We propose to generate $\mathcal{S}$ in three groups.

The first group includes patrol patterns generated from an iterative algorithm that is motivated from fictitious plays proposed by Robinson (1951), where she proved that an iterative method will generate mixed strategies that converge to the optimum in a two-person zero-sum matrix game. In that iterative method, each player chooses a pure strategy arbitrarily in the first round. In each subsequent round, each player chooses a pure strategy that produces the best expected value against the mixture of strategies used by the other player in the previous rounds. Because in our model the patroller has infinitely many strategies (patrol patterns), we first compute $(p_1, \ldots, p_n)$ based on the mixture of strategies used by the attacker in the previous rounds, and then use IPH(d) to generate a patrol pattern for the patroller. The algorithm proceeds as follows:

1. In round 1, each player picks a strategy arbitrarily.

   (a) Denote by $\xi^{(k)}$ the patrol pattern used by the patroller in round k. Choose $\xi^{(1)}$ arbitrarily.

   (b) Let the attacker pick node 1 to attack. Use r_i, $i = 1, \ldots, n$, to keep track of the number of times node i is picked by the attacker. Initialize $r_1 = 1$, and $r_i = 0$, for $i = 2, 3, \ldots, n$.

2. Repeat the following for a predetermined number of rounds. In round $k \geqslant 2$,

   (a) Set $p_i = r_i / \sum_{j=1}^{n} r_j$, which represents the attacker's mixed strategy based on his attack history from rounds 1 to $k-1$. Use the IPH(d) to generate a patrol pattern $\xi^{(k)}$.

   (b) Find the best node to attack by assuming the patroller uses patrol pattern $\xi^{(j)}$, $j = 1, \ldots, k-1$, each with probability $1/(k-1)$. If attacking node i yields the highest expected cost, then set $r_i \leftarrow r_i + 1$.

In round 1, we are free to set $\xi^{(1)}$ to any patrol pattern, but we propose the patrol pattern generated by the IPH(d), when the attack probability at node i is

$$p_i = \frac{1/(c_i E[X_i])}{\sum_{j=1}^{n} 1/(c_j E[X_j])}. \tag{26}$$

With this choice, from Equation (11) we see that $\lim_{k\to\infty} W_i(k) = c_i p_i \Lambda E[X_i]$ is the same for all i. In other words, with this attack probability distribution, the index of each node will approach the same limit if the node has not been visited for a long time, which results in a patrol pattern that is likely to cover all nodes.

In the kth round of this algorithm, the patrol pattern $\xi^{(k)}$ is the best one against the attacker, among d patrol patterns generated with look-ahead windows $1, 2, \ldots, d$. These other patrol patterns, in many cases identical to $\xi^{(k)}$, could add value to $\mathcal{S}$, so we include them as well. Because increasing the number of rounds does take time, adding these patrol patterns into $\mathcal{S}$ improves the overall performance with almost no additional cost.

Whereas this first group should give us a good mixture of patrol patterns, theoretically it is possible that some node is not covered in any of the patrol patterns generated in this first group. If node i is not covered in any of the patrol patterns in $\mathcal{S}$, then restricting the patroller to use only those patrol patterns in $\mathcal{S}$ will open the door for the attacker to attack node i for a guaranteed success. To fix this problem, we include a second group of patrol patterns in $\mathcal{S}$. The second group consists of all singleton patterns—a pattern consisting of just one node—into $\mathcal{S}$, so that every node is covered in at least one patrol pattern in $\mathcal{S}$. However, it is rarely a good idea for the patroller to spend all his time on a single node while ignoring all other nodes. This motivates a third (and final) group of patrol patterns to include in $\mathcal{S}$.

The third group that makes up $\mathcal{S}$ consists of n additional patrol patterns, with each pattern designed to cover one particular node but not necessarily confined to that node. We need Proposition 1 below.

Proposition 1. *Consider the random-attacker case discussed in §3.1, where $\mathbf{p} = (p_1, \ldots, p_n)$ with p_i denoting the attack probability at node i, $i = 1, \ldots, n$. If $X_i \geqslant 1$ for all i, and if $c_j p_j > \sum_{i \neq j} c_i p_i$, then a patrol policy that never visits node j cannot be optimal.*

The proof of this proposition is deferred to §EC.2.4. To find an attack probability distribution **p** such that the patroller has to visit node j with the optimal policy and still has incentive to visit some other nodes, we let $p_i = K/(c_i E[X_i])$ for $i \neq j$, where K is a constant. The reason to let p_i, $i \neq j$, be inversely proportional to $c_i E[X_i]$ is the same as that for Equation (26). From Proposition 1, we want to choose K so that

$$c_j\left(1-\sum_{i\neq j}\frac{K}{c_i E[X_i]}\right) > \sum_{i\neq j} c_i \frac{K}{c_i E[X_i]},$$

or equivalently,

$$K < 1\bigg/\left(\frac{1}{c_j}\sum_{i\neq j}\frac{1}{E[X_i]}+\sum_{i\neq j}\frac{1}{c_i E[X_i]}\right). \tag{27}$$

In our numerical experiments, we set $c_i = 1$ for all i, so we can interpret the output as the evasion probability. Therefore, we need $p_j = 1-\sum_{i\neq j} K/E[X_i] > 1-0.5 = 0.5$, where the inequality follows from (27) by setting $c_i = 1$ for all i. In our numerical experiments, we set $p_j = 0.51$ and use IPH(d) to generate a patrol pattern, with the choice of d to be discussed later. We add all patrol patterns generated with different look-ahead windows $1, 2, \ldots, d$ into $\mathscr{S}$, for the same reason discussed when generating the first group. The third group adds at most $n \cdot d$ additional patrol patterns into $\mathscr{S}$.

To summarize, we generate $\mathscr{S}$ in three groups, and our heuristic has two parameters r and d, both predetermined positive integers. For a graph with n nodes, we run the algorithm for $r \times n$ rounds, to generate the first group. If we run IPH(d) when generating patrol patterns in groups 1 and 3, then we can end up with at most $r \times n \times d$ patrol patterns in group 1, n in group 2, and at most $n \times d$ in group 3. The actual number, however, is usually much smaller than $rnd + d + nd$, because many of the generated patrol patterns produce identical performance. For instance, patrol patterns 1-2-1-3-2 and 2-3-1-2-1, although different, produce identical results at each node. In any case, it is straightforward to solve a two-person zero-sum matrix game using linear programming, when the attacker has n pure strategies, and the patroller can use any patrol patterns in $\mathscr{S}$. Our heuristic is optimal in a special case discussed in Theorem 4, whose proof is deferred to §EC.2.5.

THEOREM 4. *If $n = 2$, and $P(X_i = d_i) = 1$ for some positive integers d_i, $i = 1, 2$, then the heuristic policy for the strategic-attacker case is optimal—by using the patrol patterns in the second and the third groups.*

4.3. Lower Bound

We can modify the linear program in §3.3.4 to compute a lower bound for the optimal value in the strategic-attacker case. Recall from (21) that $\sum_{k=1}^{B_i} y_{ik} R_i(k)$ represents the long-run cost rate that can be avoided at node i. Hence, the expected cost if node i is attacked is

$$c_i - \frac{1}{\lambda_i}\sum_{k=1}^{B_i} y_{ik} R_i(k) = c_i\left(1-\sum_{k=1}^{B_i} y_{ik}\int_0^k P(X_i > t)\,dt\right).$$

To minimize the maximum such cost among all nodes, we can set up a linear program as follows:

$$\begin{aligned}&\min\quad z\\ &\text{subject to}\quad z \geqslant c_i\left(1-\sum_{k=1}^{B_i} y_{ik}\int_0^k P(X_i > t)\,dt\right).\\ &\qquad\qquad \text{and constraints in (15), (16), (19), (20),}\\ &\qquad\qquad\qquad\qquad\qquad\qquad \text{(22), (23), (24).}\end{aligned}$$

The additional constraints presented in §EC.3 also apply to the strategic-attacker case. In the next section, we use this lower bound to prepare Tables 4 and 5.

4.4. Numerical Experiments

This section presents numerical experiments for the strategic-attacker case. Increasing either of the two parameters of our heuristic, namely r and d, will improve the performance and take more time, so which investment brings better marginal benefit? To answer this question, we conduct experiments on 6-node complete and line graphs, and present the results in Figures 3 and 4, respectively.

As shown in Figure 3, the curve corresponding to $d = 1$ lies to the left and below that corresponding to $d = 2$ (less time and better performance). Setting $d = 3$ only makes the matters worse. In Figure 4, however, if we set $d = 1$, then the 90th percentile is still 3.18% over optimum even with $r = 20$. Setting $d = 2$ and $r = 5$, the heuristic (90th percentile) improves to 1.30% over optimum and cuts the running time from 0.51 seconds to 0.42 seconds. For 6-node line graphs, $d = 2$ appears to be the best choice, as the corresponding curve is to the left and below those corresponding to $d = 3, 4, 5$. After studying the other graph types (results not shown), we conclude that in the strategic-attacker case the depth of IPH should be set smaller than that in the random-attacker case in exchange for larger r. In the rest of this section, we set $r = 10$ and

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

One can check that $d = 1$ for complete graphs and $d \geqslant 2$ for other graph types.

To evaluate our heuristic, we examine the same scenarios as those in Table 2 and report the results in Table 4. Our heuristic performs very well across all five graph types, with both the mean and the median over 1,000 scenarios well below 1% above the optimal solutions. Even when the heuristic performs relatively poorly (i.e., 90th percentile),

Table 4. Performance of the heuristic in the strategic-attacker case, reported as percentage excess over the optimal evasion probability; the last column reports the mean of (LB − Opt)/Opt in percentage, where the lower bound is computed using the linear program in §4.3.

Graph	No. nodes	Mean	50th	75th	90th	Avg depth	Avg time (sec)	Lower bound
Complete	6	0.32	0.18	0.41	0.87	1.0	0.46	0.00
Complete	7	0.32	0.21	0.42	0.75	1.0	0.60	0.00
Line	6	0.29	0.07	0.29	0.80	2.0	0.65	−0.23
Line	7	0.33	0.13	0.39	0.83	2.0	0.88	−0.25
Line	8	0.38	0.20	0.51	0.98	2.0	1.11	−0.27
Line	9	0.27	0.15	0.36	0.66	3.0	3.14	−0.31
Line	10	0.27	0.18	0.39	0.66	3.0	4.04	−0.33
Line	11	0.33	0.22	0.43	0.70	3.0	5.05	−0.32
Line	12	0.33	0.22	0.42	0.70	3.0	6.19	−0.39
Line	13	0.34	0.24	0.45	0.75	3.0	7.56	−0.40
Line	14	0.35	0.26	0.46	0.72	3.0	8.97	−0.39
Circle	6	0.25	0.14	0.35	0.69	2.0	0.72	−0.05
Circle	7	0.42	0.28	0.61	1.00	2.0	0.93	−0.06
Circle	8	0.40	0.24	0.57	1.00	2.0	1.20	−0.08
Circle	9	0.45	0.34	0.65	0.97	2.0	1.48	−0.09
Random tree	6	0.22	0.03	0.23	0.68	2.0	0.64	−0.22
Random tree	7	0.34	0.11	0.41	0.92	2.0	0.83	−0.26
Random tree	8	0.63	0.24	0.65	1.49	2.0	1.08	−0.32
Random tree	9	0.82	0.36	0.93	1.99	2.0	1.41	−0.36
Hexagon grid	6	0.47	0.26	0.63	1.21	2.0	0.92	−0.04
Hexagon grid	7	0.58	0.34	0.80	1.38	2.0	1.39	−0.02
Hexagon grid	8	0.74	0.54	1.05	1.68	2.0	1.90	−0.05

it is just about 1%–2% above optimum. The last column shows the mean of the lower bound derived in §4.3, in terms of the percentage below optimum. The lower bound is, on average, within 0.5% of the optimum. In addition, the quality of the lower bound does not appear to degrade as n increases as in the case of random attackers. The main reason is that the linear programs that produce the lower bound in both cases allow mixed strategies, which resembles the optimal strategy against strategic attackers, whereas the optimal strategy against random attackers consists of just *one* patrol pattern.

To assess how much our heuristic improves over a naïve patrol strategy, we again consider two graph types, as in the case of random attackers. A naïve patrol strategy is moving back and forth for a line graph, and circle around all nodes for a circle graph. For $n = 6$, among the same 1,000 scenarios reported in Table 4, on average the naïve patrol produces an evasion probability 20.65% over optimum for line graphs and 19.59% over optimum for circle graphs. In either case, on average our heuristic produces an evasion probability less than 0.5% over optimum.

Table 5. Performance of the heuristic in the strategic-attacker case, reported as the percentage excess over the lower bound, which is computed using the linear program in §4.3.

	$n = 12$				$n = 18$			
Graph	Mean	50th	75th	90th	Mean	50th	75th	90th
Complete	0.78	0.59	1.09	1.73	1.29	1.24	1.66	2.12
Line	0.72	0.50	0.90	1.57	0.70	0.56	0.95	1.36
Circle	0.48	0.39	0.64	0.95	0.57	0.48	0.74	1.04
Random tree	1.43	1.02	1.87	3.04	1.57	1.30	1.90	2.76
Hexagon grid	1.25	1.10	1.68	2.30	2.46	1.85	2.74	5.18

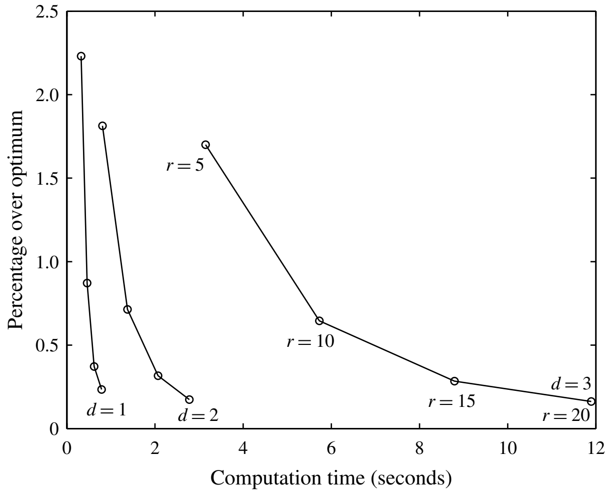
To conclude this section, we consider larger graphs and compare our heuristic with the lower bound derived in §4.3. As shown in Table 5, for 12-node graphs, on average the heuristic performs within 2% above the lower bound, and for an 18-node graph it is within 3%. The results show not only the excellent performance of the heuristic, but also the tightness of the lower bound.

5. Conclusions

In this paper, we study how to patrol a graph against random attackers and against strategic attackers. In both cases, we give an exact linear program to compute the optimal solution. Because the linear program quickly becomes computationally intractable as the problem size grows, we propose easy-to-compute index-based heuristics, which produce near-optimal performance in our numerical experiments.

Besides producing an effective patrol policy, our work can also be used to provide recommendations on how to design a patrol graph. For instance, a museum can compare patrol results to decide where its most valuable art work should be exhibited (swapping c_i and c_j) and to decide how to connect its exhibit room (adding or removing edges).

Figure 3. This figure displays, for the strategic-attacker case, the heuristic performance against computation time for different r and d, on 6-node complete graphs.

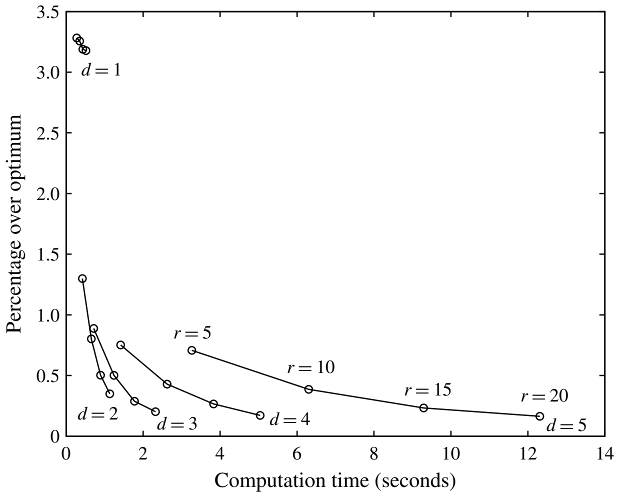


Notes. The performance is the 90th percentile over 1,000 random scenarios, reported as percentage excess over optimum. Each line corresponds to one value of d, with $r = 5$, 10, 15, 20 from left to right.

One assumption in our model is that it takes the same amount of time for the patroller to move from one node to its adjacent nodes. If a node is far away from all the other nodes, we can create *dummy* nodes in between, and let the cost (namely c_i) be zero for those dummy nodes. Another assumption we make is that the detection occurs at the end of a time period, as opposed to, say, at the beginning of a time period. Although the mathematical expressions for different variations will invariably be different, intuitively we believe the effectiveness of the proposed heuristics would be comparable for other modeling choices.

There are a few possible future research directions. First, we assume the patroller is perfect in detecting an attack. In reality, a patroller may not detect the attacker even if the two occupy the same node at the same time. Second, we study the case of one patroller. Often in practice, however, a large area is patrolled by many patrollers. Another interesting twist to our current model is to allow the attacker to see the patroller when the two occupy the same node. In that case, an attacker can initiate an attack immediately after the patroller leaves the targeted node. Another interesting extension is to develop a dynamic game, where the attacker can explore potential targets before deciding where to attack. These extensions may require substantially different formulations.

Figure 4. This figure displays, for the strategic-attacker case, the heuristic performance against computation time for different r and d, on 6-node line graphs.



Notes. The performance is the 90th percentile over 1,000 random scenarios, reported as percentage excess over optimum. Each line corresponds to one value of d, with $r = 5$, 10, 15, 20 from left to right.

Supplemental Material

Supplemental material to this paper is available at http://dx.doi.org/10.1287/opre.1120.1149.

Acknowledgments

The authors are grateful to the three referees and the associate editor for their valuable comments. This material is based upon work supported by the Office of Naval Research.

References

Alpern S, Gal S (2002) Searching for an agent who may or may not want to be found. *Oper. Res.* 50(2):311–323.
Alpern S, Gal S (2003) *The Theory of Search Games and Rendezvous* (Kluwer Academic Publishers, Norwell, MA).
Alpern S, Morton A, Papadaki K (2011) Patrolling games. *Oper. Res.* 59(5):1246–1257.
Archibald TW, Black DP, Glazebrook KD (2009) Indexability and index heuristics for a simple class of inventory routing problems. *Oper. Res.* 57(2):314–326.
Auger JM (1991) An infiltration game on k arcs. *Naval Res. Logist.* 38(4):511–529.
Baston V, Kikuta K (2004) An ambush game with an unknown number of infiltrators. *Oper. Res.* 52(4):597–605.
Baston V, Kikuta K (2009) An ambush game with a fat infiltrator. *Oper. Res.* 57(2):514–519.
Birge J, Pollock S (1989) Modelling rural police patrol. *J. Oper. Res. Soc.* 40(1):41–54.
Chaiken J, Dormont P (1978) A patrol car allocation model: Capabilities and algorithms. *Management Sci.* 24(12):1291–1300.
Chelst K (1978) An algorithm for deploying a crime directed (tactical) patrol force. *Management Sci.* 24(12):1314–1327.
Dunnigan JF (1992) *The Complete Wargames Handbook: How to Play, Design and Find Them* (Quill, Minneapolis).
Gittins JC, Glazebrook KD, Weber R (2011) *Multi-armed Bandit Allocation Indices*, 2nd ed. (John Wiley & Sons, Hoboken, NJ).
Glazebrook KD, Kirkbride C, Ouenniche J (2009) Index policies for the admission control and routing of impatient customers to heterogeneous service stations. *Oper. Res.* 57(4):975–989.
Glazebrook KD, Kirkbride C, Mitchell HM, Gaver DP, Jacobs PA (2007) Index policies for shooting problems. *Oper. Res.* 55(4):769–781.

Karp RM (1972) Reducibility among combinatorial problems. Miller RE, Thatcher JW, eds. *Complexity of Computer Computations* (Plenum, New York), 85–103.

Larson RC (1972) *Urban Police Patrol Analysis* (MIT Press, Cambridge, MA).

Lee S, Franz L, Wynne A (1979) Optimizing state patrol manpower allocation. *J. Oper. Res. Soc.* 30(10):885–896.

Olson D, Wright G (1975) Models for allocating police preventive patrol effort. *Oper. Res. Quart.* 26(4, Part 1):703–715.

Puterman ML (1994) *Markov Decision Processes: Discrete Stochastic Dynamic Programming* (John Wiley & Sons, New York).

Robinson J (1951) An iterative method of solving a game. *Ann. Math.* 54(2):296–301.

Ross SM (2010) *Introduction to Probability Models*, 10th ed. (Academic Press, San Diego).

Ruckle W (1983) *Geometric Games and Their Applications* (Pitman Advanced Publishing Program, Boston).

Shieh E, An B, Yang R, Tambe M, Baldwin C, DiRenzo J, Maule B, Meyer G (2012) Protect: A deployed game theoretic system to protect the ports of the United States. *Internat. Conf. Autonomous Agents and Multiagent Systems (AAMAS)* (International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC).

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

Thomas L, Washburn A (1991) Dynamic search games. *Oper. Res.* 39(3):415–422.

Washburn A, Wood K (1995) Two-person zero-sum games for network interdiction. *Oper. Res.* 43(2):243–251.

Washburn AR (2003) *Two-Person Zero-Sum Games*, 3rd ed. (INFORMS, Hanover, MD).

Weber R, Weiss G (1990) On an index policy for restless bandits. *J. Appl. Probab.* 27(3):637–648.

Weber R, Weiss G (1991) Addendum to "On an index policy for restless bandits." *Adv. Appl. Probab.* 23(2):429–430.

Whittle P (1988) Restless bandits: Activity allocation in a changing world. *J. Appl. Probab.* 25:287–298.

Zoroa N, Zoroa P, Fernandez-Saez MJ (2009) Weighted search games. *Eur. J. Oper. Res.* 195(2):394–411.

Kyle Y. Lin is an associate professor in the Operations Research Department at the Naval Postgraduate School. His research interests include stochastic modeling, queueing theory, game theory, and their applications.

Michael P. Atkinson is an assistant professor in the Operations Research Department at the Naval Postgraduate School. His research focuses on applying operations research techniques to military, homeland security, and healthcare applications.

Timothy H. Chung is an assistant professor of systems engineering at the Naval Postgraduate School. His research interests include modeling and analysis of operational settings involving unmanned systems, notably information gathering and sensor fusion for search and detection missions using probabilistic and optimization models.

Kevin D. Glazebrook is Distinguished Professor in Operational Research in the Department of Management Science at Lancaster University in the United Kingdom (UK). His research interest centers on the development and evaluation of heuristics for complex stochastic decision problems that are based on state-based calibrations of the options available. He directs the US$20 million LANCS Initiative whose goal is to build research capability in foundational operations research (OR) within the UK. He also chairs the STOR-i Centre for Doctoral Training in statistics and OR at Lancaster University. Both of these major national initiatives profit from significant U.S. and other international engagement.