Optimal Patrol to Defend Against Threats

Matthew Speers

February 22, 2022

Overview

Patrolling problems present themselves in many situations, often within our daily lives. When visiting a shopping centre you might see a security guard, or you could see a police patrol car on the motorway. In each case, somebody has to make a decision as to how the patroller moves around their designated area. How this decision is made often depends on the context. For example, a routine building inspection team could be assigned the schedule that maximises area coverage in the smallest amount of time possible, whereas a military patrol may wish to focus on areas that are most likely to be attacked. Even for the same applications, the scheduling method is often dependent on the organisation, with historical schedules and worker preferences usually playing a role.

Methods have been developed to automate this scheduling process for arbitrary areas. By breaking down a patrol area into its most crucial points and the connections between these points, a mathematical model can be constructed using graph theory. Each patrol point is referred to as a node, and the connections between them as edges. Designing the patrol schedule then becomes a question of how to move the patroller from node to node, along the edges. To give the patroller something to do at each node, the idea of potential attacks is introduced. These could be anything from an art thief trying to break into a museum or a cyber attacker attempting to breach a system. Sometimes these attackers are assumed to have inside knowledge of the system they are trying to attack, and thus knowledge of how the patroller is likely to move. Other times, they are assumed to have no knowledge and so just attack randomly. The patrol designer's objective is to move the patroller between the nodes in a way that maximises their attack detection (and thus prevention) potential.

Mathematical assumptions about the problem can be made to quantify the patroller's potential to detect attacks, for example by looking at how many attacks are expected to be taking place at a given node. This is arguably the simplest approach, with some approaches taking other factors into account such as how soon we can expect the ongoing attacks at a node to be completed. There is also the question as to how effective the patroller is at detecting attacks. A simple idea would be to assume they have a 100% successful detection rate, however, this may not be realistic; perhaps a shoplifter knows how to act natural in a crowd of patrons. Some authors account for this possibility by instead saying that we can only expect to detect a certain percentage of attacks.

Once all of these ideas have been combined into a mathematical model, we then need to be able to solve these models. It is often the case in practice that, for a large enough patrol area, finding the exact solutions takes an unrealistic amount of time and computing power. To overcome this, heuristic approaches have been developed that result in solutions that are a good approximation of the true optimal patrol schedule. These heuristics are much less costly to compute and so are much more suitable for use in real-world decision making, when time and cost are often important factors.

1 Introduction

There are many real-world scenarios which involve defending against attacks which occur at unknown times and locations. Take, as examples, a security guard roaming a shop floor to prevent theft or a military force protecting a strategic location. In each of these situations, the patroller seeks to design their patrol pattern in order to maximise the effectiveness of threat detection. This type of problem has links to the search problem described by Koopman (1957), in which an object is hidden in an unknown location and a finite search resource must be allocated to attempt to find it. More directly comparable developments include that of Chaiken and Dormont (1978), where they design a computer program which determines an optimal patrol schedule for police cars by minimising metrics such as total wait time and average response time. However, in this application, the attacks are assumed to be fixed in location and time. Other authors consider the problem in reverse, such as Hohzaki et al. (2013) who take the perspective of a thief trying to break into an art gallery.

There is interest in these problems from a broad range of agencies. Jiang et al. (2012) use optimal patrolling methodology to manage fare inspection in public transport. Applications to military strategy are discussed by Lin (2021), where the patroller aims to protect a perimeter against disguised and non-disguised attackers. In Haas and Ferreira (2018), strategies are devised to attempt to intercept rhino poaching parties within wildlife reserves.

Here, the focus is primarily on the work of Lin et al. (2013), with a briefer discussion of Lin et al. (2014). The former looks at the case of perfect attack detection and the latter at imperfect attack detection. In each paper they consider two situations: one where the attacks made to the system are random and another where the attacks are strategic. The authors use a graphical model of the system to represent possible attack locations and travel times for the patroller and then formulate the problem as a Markov Decision Process (MDP). The objective is to find an optimal patrol schedule around these attack locations. To do this, they develop index-based heuristics as the associated linear programs become computationally infeasible for even moderately sized problems.

2 Perfect Detection

This section details the model described by Lin et al. (2013), where it is assumed that if a patrol occupies the same area as an attack, the attack will always be detected. The potential attack area is separated into n locations, represented as the nodes in the graph. If it is possible for a patroller to travel between two locations, the corresponding nodes are connected by an edge which we say takes 1 time unit for the patroller to traverse. For example, if the patroller has access to only the perimeter of a compound we can reflect this by using a circular graph for the model. Alternatively, a connected graph could represent the case of a satellite that can transition near-instantaneously from viewing any location to another. There are many other possible graph structures (line, random tree, etc.) that can be used to reflect a variety of real-world scenarios. Here, the majority of the analysis is done for the case of the connected graph.

In the case of random attacks, Lin et al. (2013) assume the attackers arrive according to a Poisson process with rate Λ. They let each attacker select node i to attack with probability p_i, so the rate at which attacks arrive at node i is $\lambda_i = \Lambda p_i$. This is the attack model used until we consider the strategic attacker case in §2.3. If the patroller occupies the same node as an ongoing attack at the end of a time period, the attack is detected and prevented. The patroller's objective is to traverse this graph and minimise the total expected cost from attacks, where the cost of a successful attack at node i is denoted c_i.

2.1 MDP Formulation

We now outline how Lin et al. (2013) formulate the patrol problem as a Markov Decision Process. Since all attacks are detected upon a visit to a node, the state of the system can be described completely by information about the visits. We thus write $\mathbf{s} = (s_1, \ldots, s_n)$ for the state of the system, where s_i is the time since the last visit to node i. Traversal of an edge takes 1 time unit, so the state space is equal to $\Omega = \{(s_1, \ldots, s_n) : s_i \in \mathbb{Z}^+ \text{ for } i = 1, \ldots, n\}$. The action taken by the patroller is the decision of which adjacent node to move to after detection at the current node. They also have the option to remain at the current node. For the connected graph case, which we are considering here, the action space is therefore $A = \{i : i = 1, \ldots, n\}$. Any patrol policy $\pi : \Omega \to A$ is a mapping from the state space to the action space. If the patroller selects node i as an action, the state transitions deterministically with $s_i \mapsto 1$ and $s_j \mapsto s_j + 1$ for $j \neq i$. Denote this new state $\phi(\mathbf{s}, i)$, the result of taking action i in state $\mathbf{s}$.

Any attacks that complete in the next time period cannot be prevented, as the patroller only detects attacks at the end of the time period. Therefore, the expected total cost in the next time period is equal to the sum over *each node j* of the *expected number of completed attacks at j within this time period* multiplied by the *respective cost c_j*. For a random attack time X_j, Lin et al. (2013) show that the expected number of attacks that occur at j in the next time period is $\lambda_j \int_{s_j - 1}^{s_j} \mathbb{P}(X_j \leq t)\, \mathrm{d}t$. Multiplying this by c_j gives the expected cost incurred at j in the next time period, $C_j(\mathbf{s}, i)$. The total expected cost incurred in the next time period is therefore

$$C(\mathbf{s}, i) = \sum_{j=1}^{n} \left(c_j \lambda_j \int_{s_j - 1}^{s_j} \mathbb{P}(X_j \leq t)\, \mathrm{d}t \right) = \sum_{j=1}^{n} C_j(\mathbf{s}, i). \tag{1}$$

Note that the cost incurred in the next time period does not depend on the action taken. The action i is specified in (1) because it determines the state transition and so costs incurred in future time periods.

The integral in (1) is the same for any $s_j \geq B + 1$, where B is an integer upper bound for X_t. The infinite state space Ω can thus be restricted to the finite $\Omega' = \{(s_1, \ldots, s_n) : s_i \in \mathbb{Z}^+; s_i \leq B + 1 \text{ for } i = 1, \ldots, n\}$, where $s_j \mapsto \min(s_j + 1, B + 1)$ for $j \neq i$ when transitioning. Lin et al. (2013) note that, because the action space A is also finite, we only need to consider deterministic, stationary policies (Puterman, 1994).

Another reason to restrict the state space is to allow us to obtain an expression for the long-run cost rate. Denote the resultant state of applying deterministic policy π to state $\mathbf{s}$ as $\psi_\pi(\mathbf{s})$. The sequence of states resulting from applying policy π to initial state $\mathbf{s}_0$ can thus be written as $\mathcal{S}(\mathbf{s}_0, \pi) = \{\psi_\pi^k(\mathbf{s}_0), k = 0, 1, \ldots\}$. There are a finite number of possible states, so eventually one must be revisited in $\mathcal{S}(\mathbf{s}_0, \pi)$. Meaning, after some initial steps through transient states, $\mathcal{S}(\mathbf{s}_0, \pi)$ consists of a repeated cycle of states. Lin et al. (2013) therefore write the long-run cost rate at node i when policy π is applied to initial state $\mathbf{s}_0$ as

$$V_i(\pi, \mathbf{s}_0) = \lim_{N \to \infty} \frac{1}{N} \sum_{k=0}^{N-1} C_i\left(\psi_\pi^k(\mathbf{s}_0), \pi(\psi_\pi^k(\mathbf{s}_0))\right), \tag{2}$$

where $\pi(\mathbf{s})$ is the node selected by π at $\mathbf{s}$. The patroller's objective is to minimise (2) and so the optimal value is

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

where Π is the class of deterministic, stationary policies. The optimal value C^{OPT} does in fact not depend on $\mathbf{s}_0$ for the connected graph case. As is explained by Lin et al. (2013), this is because any patrol sequence can be constructed by selecting an appropriate π, regardless of the initial state.

2.2 Heuristic Policies

As mentioned in §1, it is not feasible to solve the MDP using linear programming or other standard techniques for relatively large n. The heuristic polices Lin et al. (2013) devise as an alternative are discussed here. They begin by breaking the problem into n single node problems to be solved alongside each other. To do this, the policy space Π is extended to Π^{MN}, the set of deterministic, stationary policies that can allow visits to multiple nodes within the same time period. Importantly, the constraint $\sum_{i=1}^{n} \mu_i(\pi, \mathbf{s}_0) \leq 1$, where $\mu_i(\pi, \mathbf{s}_0)$ is the rate at which node i is visited given $\mathbf{s}_0$ and π, must be satisfied. This is called the total-rate constraint and is satisfied by all policies in Π. The new objective function for the individual problem for node i is thus

$$\min_{\pi \in \Pi^{\text{MN}}} V_i(\pi) + w_i \mu_i(\pi), \tag{4}$$

where the second term comes from the Lagrangian relaxation of the total-rate constraint. In this context, w_i can be thought of as the service cost for visiting node i. We can again drop the dependency on $\mathbf{s}_0$ due to the arguments made regarding (3). Since there is only 1 node to visit in this case, all the policies that need to be considered can be represented by how often they visit this single node, say every k time periods. Since all attacks are detected upon a visit, each visit constitutes a renewal of the process and so every k time units is a cycle. By calculating the expected number of attacks within a cycle, Lin et al. (2013) show that solving (4) is equivalent to solving $\min_{k \in \mathbb{Z}^+} f(k) = c_i \lambda_i \int_0^k \mathbb{P}(X_i \leq t)\, \mathrm{d}t/k$, as $f(k)$ is the long-run cost rate at i for a policy that visits every k time periods. Using this, they define the cost w_i required for visiting i every k or $k+1$ time periods to incur the same cost rate as

$$W_i(k) = c_i \lambda_i \left(k \int_k^{k+1} \mathbb{P}(X_i \leq t)\, \mathrm{d}t - \int_0^k \mathbb{P}(X_i \leq t)\, \mathrm{d}t \right). \tag{5}$$

They interpret this as "*the maximum per visit cost for the policy that visits the node in state k (once every k time periods) to be optimal*" (Lin et al., 2013). For a given state $\mathbf{s} = (s_1, \ldots, s_n)$ a heuristic based on (5) can be implemented by selecting action i such that $i \in \{j : W_j(s_j) = \max_{k=1,\ldots,n} W_k(s_k), j = 1, \ldots, n\}$, i.e. by moving to a node i that has the highest index. Lin et al. (2013) show the IH based on (5) to be optimal for $n = 2$ and close to optimality for cases of $n \geq 3$.

The first heuristic policy devised by Lin et al. (2013) is called the *index reward heuristic* (IRH), where the index $W_i(s_i)$ of the chosen action is viewed as a reward. The patroller will calculate the sum of the indices along each possible path of length l from the current state. They then select the path with the highest aggregate and move to the node that is the first step in this path. The choice of l can change the patrol path selected, however, larger values of l do not necessarily give better selections. Therefore, it is valuable to consider the paths of lengths $1, \ldots, l-1$, as well as length l. To this end, Lin et al. (2013) define IRH(d) to be when the patroller calculates the d paths of lengths $1, \ldots, d$ with the highest aggregated IRH and selects the one which gives the patrol pattern with lowest long-run cost rate. Since more paths are being considered when d is increased, the performance of the IRH(d) improves with d. Unfortunately, the IRH(d) can perform poorly in certain situations by waiting for the IH of a node to increase so that it can collect a higher reward in the future. This can result in the patroller missing attacks they otherwise wouldn't, as is explained by Lin et al. (2013).

They introduce another heuristic, the *index penalty heuristic* (IPH), which does not encounter this problem. Rather than see the index of the chosen node as a reward, it considers the sum of the indices for the unselected nodes as a penalty. So, for each of the l steps in a potential path, the patroller calculates $\sum_{i \neq j} W_j(s_j)$, where i is the node in step i. This is then aggregated across each path and the chosen node is the first step in the path with the smallest aggregate. It is again not guaranteed that the selection made is better for larger l. The IPH(d) heuristic is thus

defined similarly to the IRH(d), where the d paths with the lowest IPH are compared and the one with the lowest long-run cost rate selected. The next action is then the first step in this path. Again, the performance of IPH(d) increases with d.

For the sake of their numerical study, Lin et al. (2013) also define a myopic heuristic (MH) which selects the node which results in the highest expected number of detected attacks upon a visit. They show this to be $R(\mathbf{s}, i) = c_i \lambda_i \int_0^{s_i} \mathbb{P}(X_i > t)\,\mathrm{d}t$, where $R(\mathbf{s}, i)$ is referred to as the reward gained by visiting node i in state $\mathbf{s}$. This can be extended to MH(d) in a similar fashion to the above. The authors also formulate a linear program to calculate a lower bound for C^{OPT} to aid their numerical evaluation.

Lin et al. (2013) conduct numerical experiments to compare the heuristics discussed above. Both IRH and IPH outperform MH, with the IPH showing a larger improvement. The authors argue this is due to the potential issue regarding IRH waiting for a higher index before visiting a node. They then assess the performance of IPH for varying d and different graph types and note that increasing d increases both the performance and computational cost. Interestingly, the relative performance increase with d is dependent on the graph structure, with the improvement being more noticeable for less connected graphs (e.g. line). They thus propose the *modified index penalty heuristic* (MIPH) which is IPH(d) with $d = 1 + \lceil$average distance between all pairs of nodes$\rceil$. This approach gives $d = 2$ for complete graphs and larger d for less connected graphs, providing a balance between keeping computational cost low and increasing d when it is most beneficial. They then compare the solutions found by MIPH to the optimal solution found via a linear program and show it to perform well, with the MIPH solutions exceeding the linear program based lower bound by an average of 2%.

2.3 Strategic Attacks

A scenario where the attacker has knowledge of the system and selects an attack node to maximise cost is also discussed by Lin et al. (2013). This turns the problem into a two-person zero-sum game and therefore has links to game theoretic approaches, for example those used by Auger (1991) and Alpern and Gal (2002). The patroller's objective becomes

$$\min_{\pi \in \Pi^{\mathrm{R}}} \max_{i=1,\ldots,n} \frac{V_i(\pi)}{\lambda_i}, \tag{6}$$

i.e. minimise the maximum long-run cost rate, or minimise the cost of the action taken by the attacker. Here Π^{R} is the set of randomised policies. Again, this can be solved by a linear program which becomes computationally intractable for reasonably large n. To overcome this, Lin et al. (2013) suggest formulating a different two-person zero-sum game in which the attacker chooses a node i to attack and the patroller takes up a mixed strategy from m patrol patterns given in $\mathscr{S} = \{\xi_1, \ldots, \xi_m\}$. They note that it is straightforward to set up a linear program to solve this problem, as is done in Washburn (2003). Importantly, this linear program is much easier to solve than the one that computes the true optimal solution. The challenge comes from choosing this set $\mathscr{S}$. Lin et al. (2013) devise an iterative algorithm which uses IPH(d) to select patrol patterns to add to $\mathscr{S}$ and is motivated by the work of Robinson (1951). This algorithm is performed $r \cdot n$ times, with the details being omitted here. They then add to $\mathscr{S}$ the set of n patrol patterns that only visit a single node, in order to ensure all nodes are visited by at least one pattern in $\mathscr{S}$. Finally, they use IPH(d) to add up to $n \cdot d$ patrol patterns which each include, but are not confined to, a given node. The two-person zero-sum game is then solved to find the heuristic policy, which is in fact optimal for a special case of $n = 2$ (Lin et al., 2013).

Numerical experiments conducted by Lin et al. (2013) show increasing r to be a better use of computing power than increasing d. They therefore define the heuristic $d = 1 + \lceil$(average distance between all pairs of nodes $- 1)/2\rceil$

which sets d smaller than in the random attacker case. By comparing to the optimal solution (for small n) and a lower bound based on a linear program (for larger n), they show this heuristic performs well, with averages of 0.5% over the optimum and up to 3% over the lower bound.

3 Imperfect Detection

The model used by Lin et al. (2014) is largely the same as the same as the one described in §2, with the crucial difference being that the patroller is no longer guaranteed to detect all ongoing attacks when visiting a node. The probability of detecting an individual attack at node i is denoted α_i and the attack time at node i is said to have cumulative density function $F_i(\cdot)$, bounded by B for all i.

3.1 Formulation as an MDP

Under the assumptions above, the time for an attack to complete can be no longer than B. For this reason, it is sufficient to describe the state of the problem at time 0 with only what has happened in the time interval $[-B, 0)$. Hence, Lin et al. (2014) define the state of the system as $\mathbf{s} = (s_1, s_2, \ldots, s_{B-1})$ where s_k is the node visited at time $-k$. This gives a state space of $\Omega = \{(s_1, \ldots, s_{B-1}) : s_k = 1, \ldots, n, \text{ for } k = 1, \ldots, B-1\}$ and deterministic state transition upon a visit to i of $s_1 = i$ and $s_{k-1} \mapsto s_k$ for $k > 1$. The action space is $A = \{i : i = 1, \ldots, n\}$ as in §2.1. Both the action and state space are finite, so we only need to consider deterministic, stationary policies $\pi : \Omega \to A$ (Puterman, 1994). Define v_{jk} to be the indicator that the patroller visited node j at time k.

The objective function for this MDP is defined by Lin et al. (2014) the same manner as (3), with the only difference being that here the expected cost due to attacks at j in the next time period, given a visit to i, is equal to

$$C_j(\mathbf{s}, i) = c_j \lambda_j \left(\int_0^1 F_j(1-t)\mathrm{d}t + \sum_{m=0}^{B-1} (1-\alpha_j)^{\sum_{k=1}^{m} v_{jk}} \int_m^{m+1} (F_j(t+1) - F_j(t))\mathrm{d}t \right) \text{ for } j \neq i,$$

$$C_i(\mathbf{s}, i) = c_i \lambda_i \left(\int_0^1 F_i(1-t)\mathrm{d}t + \sum_{m=0}^{B-1} (1-\alpha_i)^{1+\sum_{k=1}^{m} v_{ik}} \int_m^{m+1} (F_i(t+1) - F_i(t))\mathrm{d}t \right).$$

Again, this MDP can be solved by a linear program which becomes computationally intractable for moderately sized n.

3.2 Heuristic Policies for Random Attackers

Lin et al. (2014) develop several heuristics which attempt to solve this MDP. Some are derived from the Lagrangian relaxation (LR) of a multi-node version of the problem, similar to what is discussed in §2.2. Others are based on approximate dynamic programming (APD). Due to the limited scope of this report, we will only discuss the latter.

First assume that, irrespective of the decision made at the current time point, future visits to each node i follow a Poisson process with rate ν_i. Lin et al. (2014) consider a single node and node state $\boldsymbol{v} = (v_1, \ldots, v_{B-1})$, where $v_k = 1$ if the node was visited at time $-k$. They calculate the benefit of taking an action i at time 0 by comparing the expected reward if i is visited to the expected reward if i isn't visited. The action taken at time 0 only affects attackers that arrive in the interval $(-B, 0]$, as any other attacks will either have been completed or will be yet to

begin. These attacks can only be detected by patrols in $(-B, B]$, however, the decisions about patrols in $(-B, 0)$ have already been made and so we only need to consider patrol decisions in $[0, B]$. Lin et al. (2014) show that the expected number of these attacks that are ongoing at time 0, but have not been completed by time $s \in [0, B]$, is

$$\Phi(s, \boldsymbol{v}) = \lambda \left(\sum_{k=0}^{B-1} (1-\alpha)^{\sum_{i=1}^{k} v_i} \int_{k+s}^{k+1+s} \bar{F}(t) \mathrm{d}t \right), \tag{7}$$

where we drop the i subscripts since only one node is being considered here. Using (7), and by conditioning on there being ℓ patrols to the node in $[0, B]$, they show that the expected number of these attackers detected in $[0, B]$ is

$$\Psi(v, \boldsymbol{v}) = \sum_{\ell=1}^{\infty} \left(\frac{(vB)^{\ell}}{\ell!} \exp(-vB) \sum_{m=1}^{\ell} \alpha(1-\alpha)^{m-1} \int_{0}^{B} \Phi(t, \boldsymbol{v}) \frac{\ell! t^{m-1} (B-t)^{\ell-m}}{(\ell-1)! m! B^{\ell}} \mathrm{d}t \right), \tag{8}$$

given the node isn't visited at time 0. Therefore, if the patroller doesn't visit any node at time 0, the total expected reward accrued in $[0, B]$ from detecting these attackers is $\sum_{i=1}^{n} c_i \Psi_i(v_i, \boldsymbol{v}_i)$. If, however, the patroller does visit a node i at time 0, the expected accrued reward at this becomes $c_i(\alpha_i \Phi(0, \boldsymbol{v}_i) + (1-\alpha_i)\Psi(v_i, \boldsymbol{v}_i))$, giving a total expected accrued reward of $c_i(\alpha_i \Phi(0, \boldsymbol{v}_i) + (1-\alpha_i)\Psi(v_i, \boldsymbol{v}_i)) + \sum_{j \neq i} c_j \Psi_j(v_j, \boldsymbol{v}_j)$. The benefit (or increase in expected reward) when node i is visited at time 0 is thus

$$c_i \alpha_i (\Phi_i(0, \boldsymbol{v}_i) - \Psi_i(v_i, \boldsymbol{v}_i)).$$

The patroller then moves to the node for which this quantity is maximised. Lin et al. (2014) develop another heuristic based on the assumption that the future arrivals arrive in fixed intervals rather than according to a Poisson process, the details of which are omitted here.

Similar to the work in Lin et al. (2013), the heuristics in Lin et al. (2014) are improved by looking ahead by multiple time steps. Numerical experiments are conducted from which it is deemed best to run a version of both the LR and ADP heuristics, then select the pattern which gives the lowest long-run cost rate. Again, increasing the depth d of this heuristic proves more beneficial for less connected graphs. The authors thus set $d = 1 +$ ⌊average distance between all pairs of nodes⌋ to reflect this. They compare this refined heuristic to the optimal solution and a linear program-based lower bound. The heuristic performs well, exceeding the lower bound by at worst an average of 3.82% with a much smaller computational cost than is required for the optimal solution.

3.3 Modification of Approach for Strategic Attacks

Lin et al. (2014) consider the strategic attacker case in a manner similar to §2.3. The problem becomes a two-person zero-sum game where the attacker selects an action i for $i = 1, \ldots, n$ and the patroller selects pure strategies from a patrol pattern set $\mathscr{S}$. Here, the authors use a similar method to that used in Lin et al. (2013) to generate $\mathscr{S}$, this time using the heuristics discussed in §3.2. They generate $\mathscr{S}$ in four groups: n patterns that cover just a single node, 1 pattern that covers all nodes well, n patterns that cover a single node well, and additional patterns generated by an iterative algorithm. The details of this process are omitted here, although we mention that this new iterative algorithm attempts to improve upon the one used in Lin et al. (2014) by reducing the number of replicate patrol patterns. Once this set $\mathscr{S}$ is generated, the patroller's heuristic policy is obtained by solving the two-person zero-sum game. Numerical experiments in Lin et al. (2014) show this heuristic to be within 1% of optimally on average, with significant reductions in computation time.

4 Conclusion

In Lin et al. (2013), the authors use heuristic policies to design patrol schedules when attack detection is perfect They go on to extend this to imperfect detection in Lin et al. (2014). They develop heuristics for both cases, since solving the problems via linear programming is infeasible, and use numerical experiments to show their heuristics perform well. The success of these approaches helps to justify the use of their methodology in real-life applications.

Considerations can be made as to how to extend their model. For example, what if the attacker's behaviour changes over the course of the day? This question motivates modelling the random attacker case with a time-inhomogeneous Poisson process, rather than a homogeneous one. All of the analysis discussed here assumes one lone patroller; how large must a protection area be before it becomes beneficial to introduce another? The potential to design an area to aid patrolling also comes into question. There may be some trade off between connecting as many target areas as possible and ensuring a building design is realistic. Perhaps, by applying the methodology discussed here, it could be determined how to best balance these factors.

References

Alpern, S. and Gal, S. (2002). Searching for an agent who may or may not want to be found. *Operations Research*, 50(2):311–323.

Auger, J. M. (1991). An infiltration game on k arcs. *Naval Research Logistics (NRL)*, 38(4):511–529.

Chaiken, J. M. and Dormont, P. (1978). A patrol car allocation model: Capabilities and algorithms. *Management science*, 24(12):1291–1300.

Haas, T. C. and Ferreira, S. M. (2018). Optimal patrol routes: interdicting and pursuing rhino poachers. *Police Practice and research*, 19(1):61–82.

Hohzaki, R., Morita, S., and Terashima, Y. (2013). A patrol problem in a building by search theory. In *2013 IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA)*, pages 104–111.

Jiang, A. X., Yin, Z., Johnson, M. P., Tambe, M., Kiekintveld, C., Leyton-Brown, K., and Sandholm, T. (2012). Towards optimal patrol strategies for fare inspection in transit systems. In *2012 AAAI Spring Symposium Series*.

Koopman, B. O. (1957). The theory of search: Iii. the optimum distribution of searching effort. *Operations research*, 5(5):613–626.

Lin, K. Y. (2021). Optimal patrol of a perimeter. *Operations Research*.

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

Lin, K. Y., Atkinson, M. P., and Glazebrook, K. D. (2014). Optimal patrol to uncover threats in time when detection is imperfect. *Naval Research Logistics (NRL)*, 61(8):557–576.

Puterman, M. L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, page 451. John Wiley & Sons, Inc.

Robinson, J. (1951). An iterative method of solving a game. *Annals of mathematics*, pages 296–301.

Washburn, A. R. (2003). *Two-person Zero-sum Games*, pages 33–35. INFORMS.