Second Year Report

Thomas Lowbridge
School of Mathematical Sciences
University of Nottingham

June 13, 2018

Contents

1 Literature Review

1.1 Overview of Search games

Due to the high dependency on graphical structure, we provide a comprehensive guide to graph definitions in Appendix A.

1.2 Review of Strategic Patrolling games

A patrolling game $G = G(Q, T, m)$ is a win-lose, zero-sum game between a maximizing patroller (often referred to as she) and a minimizing attacker (often referred to as he). The parameters of the game are:

- The graph $Q = (N, E)$ made of nodes, N ($|N| = n$), joined by edges, E, which can be represented by an adjacency matrix, A.
- The length of time over which the game takes place, the time-horizon T.
- The length of time the attack takes to complete, the attack-time m.

Two forms of the game exist: the one-off game, which is played in a finite time interval $\mathcal{T} = \{0, 1, ..., T-1\}$ denoted using G^o; and the periodic game, which is played on the time circle $\mathcal{T}^* = \{0, 1, ..., T-1\}$ (with the asterisk representing arithmetic on the time circle taking place modulo T) denoted using G^p. We will assume that $T \geq m$, otherwise it clear that all attacks will fail.

The pure strategies available to the patroller are called patrols, choosing a starting position and how to walk along the graph Q, $W : \mathcal{T} \to N$. With no restrictions in the one-off game, but the restriction that the edge $(W(T-1), W(0)) \in E$ in the periodic game (so that $W(T) = W(0)$). Let

$$\mathcal{W} = \{W \,|\, W : \mathcal{T} \to N \text{ s.t } (W(t), W(t+1)) \in E \text{ for } t = 0, ..., T-2\}$$

be the set of all pure patrols in the one-off game (and similarly $\mathcal{W}^*$ in the periodic game). Let there be some ordering to the strategies $W_k \in \mathcal{W}$ (or $W_k \in \mathcal{W}^*$) for $k = 1, ..., |\mathcal{W}|$ (or $k = 1, ..., |\mathcal{W}^*|$ in the periodic game).

The pure strategies available to the attacker are pairs, $[i, I]$ for $i \in N$, called the attack node, and $I = \{\tau, \tau+1, ..., \tau+m-1\} \subseteq \mathcal{T}$ (or $I \subseteq \mathcal{T}^*$ if periodic) called the attack interval (starting at time τ). Let $\mathcal{A} = \{[i, I] \,|\, i \in N, I \subseteq \mathcal{T}\}$ be the set of all possible pure attacks. Let there be some ordering to the strategies $A_k \in \mathcal{A}$ for $k = 1, ..., |\mathcal{A}|$.

A patrol, W, intercepts the attack, $[i, I]$, if $i \in W(I) = \{W(\tau), W(\tau+1), ..., W(\tau+m-1)\}$ and as our game is Win-Lose the pure payoff function is

$$P(W, [i, I]) = \begin{cases} 1 \text{ if } i \in W(I), \\ 0 \text{ if } i \notin W(I). \end{cases}$$

A pure payoff matrix $\mathcal{P} = (P(W_i, A_j))_{i\in\{1,\ldots,|\mathcal{W}|\},j\in\{1,\ldots,|\mathcal{A}|\}}$ (with the change of $\mathcal{W}$ to $\mathcal{W}^*$ if in the periodic game) stores Win (1) or Lose (0) for each pair of pure strategies.

Let Π_W be the set of mixed strategies for the patroller in the one-off game and Π^*_W in the periodic game. Let Φ be the set of mixed strategies for the attacker.

In the mixed strategy game the patroller selects a strategy $\boldsymbol{\pi} \in \Pi_W$ (or $\boldsymbol{\pi} \in \Pi^*_W$ in the periodic game).

The attacker selects a strategy $\boldsymbol{\phi} \in \Phi$. Then the mixed payoff function (Probability of Capture) is

$$P(\boldsymbol{\pi}, \boldsymbol{\phi}) = \sum_{i=1}^{|\mathcal{W}|}\sum_{j=1}^{|\mathcal{I}|} \mathcal{P}_{i,j}\boldsymbol{\pi}_i\boldsymbol{\phi}_j = \boldsymbol{\pi}\mathcal{P}\boldsymbol{\phi}$$

(with the change of $\mathcal{W}$ to $\mathcal{W}^*$ if we are playing the periodic game).

We will also use the convention that a pure strategy is in the mixed strategy set, $W_i \in \Pi_W$ (or $W_i \in \Pi_{W^*}$) and $A_j \in \Phi$, to mean $\boldsymbol{\pi}_k = \begin{cases} 1 \text{ if } k = i, \\ 0 \text{ if } k \neq i. \end{cases}$ and $\boldsymbol{\phi}_k = \begin{cases} 1 \text{ if } k = j, \\ 0 \text{ if } k \neq j. \end{cases}$ respectively

The value of the game is denoted by $V = V(Q, T, m) \equiv \max_{\boldsymbol{\pi}\in\Pi}\min_{\boldsymbol{\phi}\in\Phi} P(\boldsymbol{\pi}, \boldsymbol{\phi}) = \min_{\boldsymbol{\phi}\in\Phi}\max_{\boldsymbol{\pi}\in\Pi} P(\boldsymbol{\pi}, \boldsymbol{\phi})$ and when needed we distinguish between the one-off and periodic game by using the subscripts V^o and V^p respectively.

Many general properties for the game can be easily proven; such as that the one-off game is non-increasing in T (as it simply increases the size of $\mathcal{I}$) and introducing more edges for the same node set doesn't lower the value (it only increases the choice in the set $\mathcal{W}$ and $\mathcal{W}^*$). Also as $\mathcal{W}^* \subset \mathcal{W}$, it is obvious that $V^p(Q, T, m) \leq V^o(Q, T, m)$ (see Alpern et al. [2011] for details). So solving the one-off game gives an upper bound for the periodic game.

We shall now focus on the unrestricted, one-off game for the rest of this report.

1.2.1 Bounds and Tools

We shall provide a list of attacks and patrollers to give bounds on V which can be applied to all graphs. The lower bounds are given in terms of the patroller's "good" strategy against all attacker options, similarly the upper bounds are given in terms of the attacker's "good" strategy against all the patroller options. When we reach tightness between the bound these "good" strategies become an optimal solution.

General bounds(Patroller and Attacker):

By the patroller waiting a random node they can achieve $V \geq \frac{1}{n}$ and by the attacker picking a random node with a fixed time I they can achieve $V \leq \frac{m}{n}$ (More generally $V \leq \frac{\omega}{n}$ for ω, the maximum number of nodes any patrol can cover).

Lemma 1.1 (General bounds)**.**

$$\frac{1}{n} \leq V \leq \frac{\omega}{n} \leq \frac{m}{n}$$

Where ω is the maximum number of distinct nodes that can be visited in an attack interval.

Decomposition(Patroller):

We can consider decomposing the graph so that we just operate on parts with some appropriate probability.

Lemma 1.2 (Decomposition lower bound)**.** *Consider decomposing Q into edge-preserving subgraphs Q_i for $i = 1, ..., k$ with values $V_i = V(Q_i)$ such that $Q = \bigcup_{i=1}^{k} Q_i$ then*

$$V \geq \frac{1}{\sum_{i=1}^{k} \frac{1}{V_i}}$$

Furthermore in the case of a disjoint decomposition equality is reached

More explicitly an edge-preserving subgraph is a subgraph who has all possible connection between its nodes and disjoint means both edge and vertex disjoint. This means we are really only selecting nodes for the subgraph and the edges are mandated.The above provides a solution to build disjointly decomposable graphs, so it is only worth studying connected graphs.

Example 1.3. For Q as seen in Figure **??**. Consider when $m = 3$, the decomposition of Q into the graphs Q_1 and Q_2 (as in Example Figure 1.1). $V_1 = V(Q_1) = 1$ as alternating between 1 and 2 can catch every attack. $V_2 = V(L_3) = \frac{3}{4}$ (as seen in Alpern et al. [2011]).

Then we can get the bound $V \geq \frac{1}{(\frac{1}{1}+\frac{3}{4})} = \frac{4}{7}$.

Simplification(Patroller and Attacker):

Definition 1.4 (Node Identification)**.** The operation of Node identification on two nodes, u and v, of a graph, $G = (N, E)$ into a single node w, is a mapping $f : N \to N'$ resulting in a new graph $G' = (N', E')$ where $N' = (N \backslash \{u, v\}) \cup \{w\}$ with $E' = E \setminus \{(u, v)\}$ if $(u, v) \in E$ and under the condition that $\forall x \in N$, $f(x) \in N'$ is incident to $e' \in E'$ iff $e \in E$ is incident to $x \in N$. Furthermore if a graph, Q, undergoes repeated node identification to become Q' then we say it has been simplified.

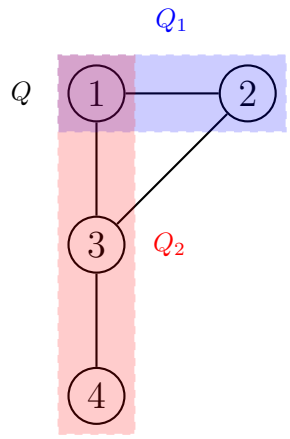


Figure 1.1: Decomposition of Q into Q_1 and Q_2.

Definition 1.5 (Embedded walk)**.** An *Embedded walk*, W', on Q' is the walk, W, done on Q under the simplification mapping of Q to Q'. i.e if $\pi : Q \to Q'$ is the simplification map, then $W' = \pi(W)$.

Lemma 1.6 (Simplification)**.** *If Q' is a simplified version of Q then $V(Q') \geq V(Q)$*

This allows us to get bounds for both the patroller and attacker.

Example 1.7. For Q as seen in Figure **??**. Consider when $m = 3$, the Simplification of the graph by identifying $1, 2$ from Q to $Q' = L_3$ (as seen in Example Figure 1.2). Hence we can get the bound that $V(L_3) \geq V(Q)$ hence $V(Q) \leq \frac{3}{4}$

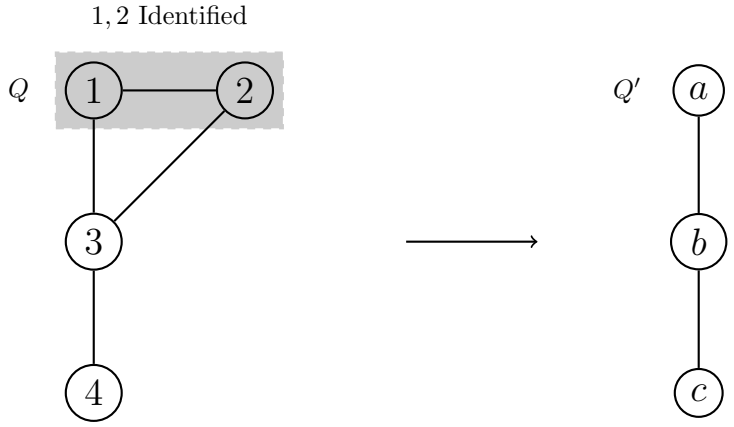


Figure 1.2: Simplifcation of Q to Q' by identification.

Diametric attack(Attacker)

Let $d(i, i')$ is the distance between nodes i and i' with the distance measured by the minimum number of edges.

Definition 1.8 (Graph Diameter)**.** The diameter of a graph Q is definded by $\bar{d} = \max_{i,i' \in N} d(i, i')$. The node pairs satisfying this are called diametrical.

Lemma 1.9. *By the attacker playing equally likely at a pair of diametrical at a random time interval, called the diametric attack, gives* $V \leq \max\left\{\frac{m}{2\bar{d}}, \frac{1}{2}\right\}$

However the bound presented in [Alpern et al., 2011] seems to indicate for large T, m the second is chosen. However a simple counter example will show the bound does not allows hold

Example 1.10 (Problem with diametric attack)**.** Consider the graph L_5 so $\bar{d} = 4$ for $T = m = 5$, then under the diametric attack, the patroller performing the patrol $\{1, 2, 3, 4, 5\}$ allows her to catch the two attacks. Hence the bound of $V \leq \frac{5}{8}$ given by lemma does not seem to hold.

Covering(Patroller) and Independence(Attacker):

Definition 1.11 (Covering)**.** A patrol, W, is called *intercepting* if it intercepts every possible attack at every node contained in the patrol, i.e all nodes visited by W are in any subpath of length m (i.e visits are at most m apart).

A set of intercepting patrols forms a *Covering set* if every node in Q is contained in at least one of the patrols. Furthermore the *Covering number*, $\mathcal{C}$ is the minimum cardinality of all the covering sets.

Definition 1.12 (Independence)**.** Two nodes, i and i', are called independent (under attack time, m) if any patrol intercepting an attack at i cannot also intercept an attack at i'.

For the one-off game this is equivalent to $d(i, i') \geq m$.

For the Periodic game this is equivalent to $d(i, i') \geq m$ and $2d(i, i') \leq T$(due to returning to start).

A set of independent points forms a *Independent set* if every element of the set is independent of every other element. Furthermore the independence number $\mathcal{I}$ is the maximum cardinality of all the independent sets.

Clearly $\mathcal{I} \leq \mathcal{C}$ as to cover a collection of independent nodes, at least that many covering patrols are needed (Possibly more if they also don't get every node in Q)

Lemma 1.13 (Covering and Independence)**.**

$$\frac{1}{\mathcal{C}} \leq V \leq \frac{1}{\mathcal{I}}$$

1.2.2 Solved Graphs

We shall provide some information on graphs that have already been solved

Hamiltonian:

A Hamiltonian graph is a graph with a Hamiltonian cycle(See Appendix [Blank] for a formal definition). Two simple examples of such graphs are cyclic graphs, C_n and the complete graph, K_n. While Hamiltonian graphs can exhibit more than one Hamiltonian cycle we shall assume that we have selected one. We shall also assume that the attack, $m < n$, as otherwise by following the Hamiltonian cycle we guarantee capture (i.e for $m \geq n \implies V = 1$).

Definition 1.14 (Random Hamiltonian Patrol)**.** A *Random Hamiltonian Patrol* is a mixed strategy starting with equal probability at all nodes and following the Hamiltonian cycle.

Theorem 1.15 (Hamiltonian)**.** *If Q is Hamiltonian, by following the Random Hamiltonian Patrol (if feasible), the patroller can achieve $V \geq \frac{m}{n}$.*

This, along with a general upper bound from [Alpern et al., 2011], provides the solution $V = \frac{m}{n}$.

Line:

A Line Graph, L_n, is a graph consisting of n nodes and $n-1$ edges connected in a straight line. While the line graph is complicated it has been solved across; Patrolling Games Alpern et al. [2011] and Patrolling a Border Papadaki et al. [2016]. The cases in the solution require the division of the (n, m) space into sub-regions (inside which different strategies are adopted by the attacker and patroller).

However upon investigation one of the sub-regions and strategies does not produce the correct bound as reported. We bring this up as in section 2.1 we will provide a counter-example to show the error and provide an altered strategy giving the required bound.

Bipartite:

A bipartite graph is a graph which can be partitioned into two sets, A and B (with $|A| = a, |B| = b$, assume WLOG that $b \geq a$) were edges only exist between these two sets. A special bipartite graph is the complete bipartite graph $K_{a,b}$.

Assume that $m < 2b$, as otherwise there exists a $2b$ period patrol which covers all nodes and guarantees capture (i.e if $m \geq 2b \implies V = 1$).

Definition 1.16 (Bipartite Attack)**.** The *Bipartite Attack* selects nodes equiprobably from the larger set B for a fixed time interval, I (or for the two time intervals, I and $I+1$ equiprobably).

Theorem 1.17 (Bipartite)**.** *If Q is bipartite with $b \geq a$, by following the Bipartite Attack, the attacker can achieve $V \leq \frac{m}{2b}$*

The reasoning behind the bound is that any patrol must alternate between $|A|$ and $|B|$, so only visits a node from B every other time step.

Corollary 1.18 (Complete Bipartite)**.** *The value of the complete bipartite graph, $K_{a,b}$, with $b \geq a$, then $V = \frac{m}{2b}$*

This is because a lower bound of $V \geq \frac{m}{2b}$ is given by the random Hamiltonian patrol in $K_{b,b}$, which simplifies to $K_{a,b}$.

1.3 Patrolling games with random attackers

This section summarizes the key work from Lin et al. [2013] and their work on Patrolling games with random attackers. Providing a baseline for future work done in Section 4

1.3.1 Game Set-up

A patrolling game is random attackers, $G = G(Q, \boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{c})$ is a minimizing game for the patroller. The parameters of the game are

- The graph $Q = (N, E)$ made of nodes, N ($|N| = n$), joined by edges, E, which can be represented by an adjacency matrix, A.
- A vector of attack time distributions, $\boldsymbol{X} = (X_1, ..., X_n)$.
- A vector of poisson arrival rates, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)$.
- A vector of costs, $\boldsymbol{c} = (c_1, ..., c_n)$

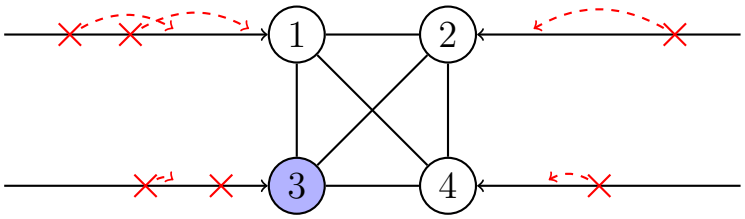


Figure 1.3: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

The attackers arrive at node, i, according a poisson process of rate λ_i, beginning their attack immediately, which lasts X_i time. The patroller detects all ongoing attacks when arriving at node i, the patroller then moves taking unit time to arrive at the next node (which can be the current node).

We can formulate such as problem as a Markov Decision Process(MDP) with state space, $\Omega = \{\boldsymbol{s} = (s_1, ..., s_n) \,|\, s_i = 1, 2, ... \text{ for } i = 1, ..., n\}$, where s_i denotes

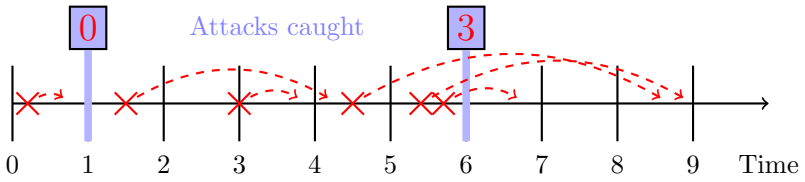


Figure 1.4: Example for a given node, when the patroller visits at times, $1, 5$

the time period since the decision to last visit node i was taken. Because the patroller can only visit one node per time period, all s_i have distinct values. In particular, one s_i, the current node, has value 1. We can identify the current node by $l(\boldsymbol{s}) = \text{argmin}_i\, s_i$. The available decisions from state $\boldsymbol{s}$ are $\mathcal{A}(\boldsymbol{s}) = \{j | A_{l(\boldsymbol{s}),j} = 1\}$ and when node i is chosen by the patroller the state transitions to $\phi(\boldsymbol{s}, i) = \widetilde{\boldsymbol{s}}$, where $\widetilde{s}_i = 1$ and $\widetilde{s}_j = s_j + 1\ \forall j \neq i$.

Because the future of the process is independent of its past, it is only the current state that matters, we can formulate its movement as a Markov Chain(MC) and hence the patroller's problem is a MDP.

The patroller incurs costs for all attackers able to complete their attacks. For the decision the cost in the next time period is the sum of all costs incurred at all nodes, I.e $C(\boldsymbol{s}, i) = \sum_{j=1}^{j=n} C_j(\boldsymbol{s}, i)$, where $C_j(\boldsymbol{s}, i)$ is the cost at node j choosing to move to node i in the next time period. Hence

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

Note. $C_j(\boldsymbol{s}, i)$ is not dependent on i, the choice of i affects the future state (and hence future incurred costs)

With a countable infinite state space, Ω, problems of finding an optimal policy may exisit (See [Need reference from Peuterman]), so we bound the state space to make it finite. A reasonable assumption is to bound the attack times and define, the smallest interger for this bound

$$B_j \equiv \min\{k \,|\, k \in \mathbb{Z}^+, P(X_j \leq k) = 1\} \tag{1}$$

We now see that the cost function remains constant, $c_j \lambda_j$, for all $s_j \geq B_j + 1$ and so we restrict our state space to $s_j \leq B_j + 1$ and modify the transitions slightly so $\widetilde{s}_j = \min(s_j + 1, B_j + 1)\ \forall j \neq i$.

Now we can consider the objective function for our MDP, we wish to minimize the long-run average cost incurred. We also know by [reference to peuterman],

that we can just focus on the class of stationary, deterministic policies $\Pi = \{\pi : \Omega \to N \,|\, \pi(\boldsymbol{s}) \in \mathcal{A}(\boldsymbol{s})\}$. So we wish to solve

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

Where $V_i(\pi, \boldsymbol{s}_0)$ is the long-run average cost incurred at node i under the policy, π, starting from state, $(\boldsymbol{s}_0)$ defined by ,

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

Where $\phi_\pi^k(\boldsymbol{s}_0)$ is the state after k transitions starting from $(\boldsymbol{s}_0)$ under the policy π.

Because the transitions are deterministic and the state space is finite, we know that $\phi_\pi^k(\boldsymbol{s}_0)$ will repeat and induce a cyclic behaviour under the policy,π. We will define the patrol pattern to be exactly this, from $\boldsymbol{s}_0$ let $\boldsymbol{s}_R$ be the first state which is repeated then define $\psi_\pi^k = \phi_\pi^k(\boldsymbol{s}_R)$, so the patrol pattern is $\{\psi_\pi^k \,|\, k = 0, 1, ..., K-1\}$. We say the patrol pattern is of length K. We can rewrite the long-run average cost at a node to be

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

We will assume we are dealing with a connected graph, otherwise we solved the connected parts separately. Therefore, because every state is reachable we see that $C^{\mathrm{OPT}}(\boldsymbol{s}_0)$ does not depend on the initial state and so $C^{\mathrm{OPT}} = C^{\mathrm{OPT}}(\boldsymbol{s}_0)$ is the same for all initial states.

Note. If we set $c_i = 1 \; \forall i$ then we can interest the long-run average cost, as the probability of not detecting an attack.

We can now use standard techniques such as value iteration or linear programming to compute the optimal policy and long-run average cost. This is left to Appendix B. However such methods are slow are only computable for a small graph, therefore we seek to create a near-optimal heuristic policy that can run in a shorter space of time.

1.3.2 Problem relaxation

We first relax the problem of the patroller only being able to visit one adjacent node each time period, to the *Multi-Node*(MN) problem, where the patroller can

visit multiple nodes each time period. We will denote the class of stationary, deterministic policies as

$$\Pi^{\text{MN}} = \{\pi^n : \Omega \to \boldsymbol{\alpha} \,|\alpha_i = 0, 1 \text{ for } i = 1, ..., n\}$$

For ease of notation we will define $\pi_i : \Omega \to \alpha_i \in \Pi_i^{\text{MN}}$, that is the resultant element map, which we will use when we only care about a single node.

Note. Our previous un-relaxed policies, $\pi : \Omega \to N$ can be converted to a MN policy by mapping $\boldsymbol{s}$ to $\boldsymbol{\alpha}$ where $\alpha_i = 1$ for $i = \pi(\boldsymbol{s})$ and $\alpha_i = 0$ for $i \neq \pi(\boldsymbol{s})$ to form a policy $\pi^n \equiv \pi$.

Like before π^n will induce a patrol pattern, $\psi^k_{\pi^n}$ with length K'. We define the long-run average visit rate at which node, i is visited to be under π^n starting from $\boldsymbol{s}_0$ as

$$\mu_i(\pi_i, \boldsymbol{s}_0) = \frac{1}{K'} \sum_{k=0}^{K'-1} \pi_i(\psi^k_{\pi^n})$$

Now we restrict ourselves to having a maximum long-run average visit rate of one. This is know as the *Total-Rate*(TR) constraint. So we restrict ourselves to obey this constraint and the set of policies,

$$\Pi^{\text{TR}} = \left\{ \pi^n \in \Pi^{\text{MN}} \,\middle|\, \sum_{i=1}^{N} \mu_i(\pi_i, \boldsymbol{s}_0) \leq 1 \,,\, \forall \boldsymbol{s}_0 \in \Omega \right\}$$

We define the Problems optimal minimum cost, $C^{\text{TR}} = \min_{\pi^n \in \Pi^{\text{TR}}} \sum_{i=1}^{n} V_i(\pi^n)$, similarly to before.

Note. We have dropped the dependency on the initial state, $\boldsymbol{s}_0$, as even though $V_i(\pi^n, \boldsymbol{s}_0)$ depends on it, the long-run average cost does not. We will similarly drop the notation on the long-run visit rate, using $\mu_i(\pi^n)$.

Again, $\exists\, \pi^n \in \Pi^{\text{TR}}$ s.t $\pi \equiv \pi^n$, so $C^{\text{OPT}} \geq C^{\text{TR}}$

Secondly we relax the MN problem by introducing the TR constraint as a Lagrangian multiplier, $\omega \geq 0$, to form

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

This formulation means that $C^{\text{TR}} \geq C(\omega)$, hence $C^{\text{OPT}} \geq C^{\text{TR}} \geq C(\omega)$.

The Lagrangian relaxtion can be interpreted as the cost of missed attacks plus a cost of ω for ever visit we make, therefore we call ω the *service charge*. As the Lagrangian relaxation, splits up into n separate problems, where node i's problem is

$$C_i(\omega) \equiv \min_{\pi_i \in \Pi_i^{\text{MN}}} \left(V_i(\pi_i) + \omega \mu_i(\pi_i)\right)$$

With $C(\omega) = \left(\sum_i^n C_i(\omega)\right) - \omega$

1.3.3 Single-node problem

Focusing on the separated, single node problem is equivalent to deciding when to visit the node. We shall remove node subscript, i, for ease of reading. Each time has a binary decision, visit (1) or wait (0). Because we are only considering stationary, deterministic policies, the optimal action of when to visit remains constant and the optimal policy will be one that visits every, k periods. Such a policy gives us an expected number of arrivals who finish before we visit of

$$\lambda \int_0^k P(X \leq k - t)dt = \lambda \int_0^k P(X \leq t)dt$$

Each successful attack costs c and the visit costs us ω and this cycle is of length k so the long-run average cost is

$$f(k) \equiv \frac{c\lambda \int_0^k P(X \leq t)dt + \omega}{k}$$

Thus solving the single node problem is equivalent to minimizing $f(k)$, by setting $f(k) = f(k+1)$ we can find the cost that makes the patroller indifferent between visiting every k and $k+1$ time units. This solution helps us characterise the optimal policy that minimizes $f(k)$ defined as

$$W(k) \equiv c\lambda \left(k \int_k^{k+1} P(x \leq t)dt - \int_0^k P(x \leq t)dt\right)$$

We have $W(0) = 0$ and as X is bounded by B, we have for $k \geq B$ that $W(k) = c\lambda E[X]$. We us this function to characterise the optimal policy minimizing the single node objective

Theorem 1.19 (Single node optimal policy)**.** *a) $W(k)$ is non-decreasing in k.*

b) If $\omega \in [W(k-1), W(k)]$ then it optimal to visit the node once every k time periods, for $k = 1, 2, ...$

c) Moreover if $\omega \geq c\lambda E[X]$ it is never optimal to visit the node.

Proof: See Appendix B

This motivates a simple Index Heuristic(IH) based on the optimal solution to the single node problem, by reinserting the node subscript, we suggest an index of

$$W_i(\boldsymbol{s}) \equiv c_i \lambda_i \left(s_i \int_{s_i}^{s_i+1} P(X_i \leq t) dt - \int_0^{s_i} P(X_i \leq t) dt \right)$$

The IH computes the index for all adjacent nodes (including the current node) from the state $\boldsymbol{s}$ and moves to the node with the highest index. In the cases of ties in indices, they are broke arbitrarily.

1.3.4 More heuristics

The IH is simplistic and short sighted, we can develop more sophisticated heuristics based on the index.

Index Reward Heuristic(IRH) We can interpret the index as a reward for visiting the node and pick a look-ahead window of length l, we then look at all paths of length l from the current node, aggregating the index along the path. Then choosing the next node to visit according to which path has the highest aggregate reward.

Index Penalty Heuristic(IPH) We could also interpret the index as a penalty for not visiting the node. With a look-ahead window of length l, we look at all paths of length l from the current node, aggregating the indices not along the path. Then choosing the next node to visit according to which path has the lowest aggregate penalty.

Note. We only use the aggregate reward/penalty to determine the next node, then we repeat the process, we do not follow the path.

A natural question to ask is; does increasing the look-ahead window improve the heuristic. The answer is no, as l and $l+1$ may return two distinct patrol patterns, with l's patrol pattern performing better than $l+1$'s. However when considering using a look-ahead window of $l+1$ we have to compute l's paths, so we might as well look at all look-ahead windows up to a value. We shall call this the heuristic depth, d and denote the depth heuristics as IRH(d) and

IPH(d) which apply IRH and IPH respectively to all look-ahead windows of length $l = 1, ..., d$.

Note. IRH(1) $\equiv$ IPH(1)

The numerical results of these heuristics can be found in Section 3.4 in [Lin et al., 2013]. A key result of the study, that we will look at later is that IPH seems to outform IRH on the Complete and Line graph.

2 Strategic Patroller and Strategic Attacker

2.1 Problem and correction to line graph strategy

Consider the patroller's strategy against a diametric attack, that simply oscillates between the two diametric points.

The total number of attacks the attacker is making under this diametric strategy is $2(T - m + 1)$, we will now measure how many the simple strategy for the patroller gets.

We will divide the set of captured attacks, depending on what is happening. This division shall be into start captures, middle captures and end captures.

The start captures are captures catching less than m attacks in the early times, i.e before the middle. The middle captures are captures catching exactly m attacks. The end captures are captures catching less than m attacks in the late times.

Example 2.1 (Problem with diametric attack)**.** Consider the graph L_5 so $\bar{d} = 4$ for $T = 20$,$m = 6$, then under the diametric attack, the patroller oscillating between diametrics points gets.

Start Capture $1 + 5 = 6$ attacks initially.

Middle Capture $6 + 6 = 12$ attacks when arriving at node 5.

End Capture 4 attacks when finishing at node 1.

Giving 22 out of $2(20 - 6 + 1) = 30$ attacks, a better than $\frac{3}{4}$ value.

First off it is worth considering that the number of end attacks that are going to be caught will come from two values (if more than these would be in the middle). We could suggest that waiting at the start is more preferable to "stabilize" into the middle quicker. the cost for doing so is to remove one from each end value, if one of the end values is 1, then the penultimate middle is also reduced by 1 and thus becomes an end value.

While each time we decide to wait gains us 1 for each node in the start values, until it becomes a middle value. As $m \geq \bar{d}$, we are guaranteed that there are at least two start values (as we are looking at times 0 and $\bar{d}-1 < m$). Therefore it is certain that waiting at the start (at least while there are two start values) is not worse than the just oscillating strategy.

Therefore we can wait until $t = m - (\bar{d}-1)$ (which as $m > \bar{d}-1$ means its is always possible), then this is only the start.

Let us count the pattern under this strategy,

Start: Capture $m - \bar{d}$ attacks initially by waiting.

Middle: Capture $m \times (\lfloor \frac{T-2m+1}{\bar{d}} \rfloor + 1)$ attacks in the middle cycles (if any middle times are possible, otherwise zero if negative).

End: Capture $T - 1 - (m - 1 + (\lfloor \frac{T-2m+1}{\bar{d}} \rfloor + 1)\bar{d})$ (at the penultimate node visit (if possible, this really is zero if its negative) and $T - 1 - (m - 1 + (\lfloor \frac{T-2m+1}{\bar{d}} \rfloor + 2)\bar{d})$ at the final node visit (again zero if negative).

This gives

$$m - \bar{d} + \left(m \times (\left\lfloor \frac{T-2m+1}{\bar{d}} \right\rfloor + 1)\right)_+ + \left(T - (m - 1 + (\left\lfloor \frac{T-2m+1}{\bar{d}} \right\rfloor + 1)\bar{d})\right)_+ + \left(T - (m - 1 + (\left\lfloor \frac{T-2m+1}{\bar{d}} \right\rfloor + 2)\bar{d})\right)_+$$

We will call $\alpha = \lfloor \frac{T-2m+1}{\bar{d}} \rfloor$

Lemma 2.2 (Condition on T for bound to hold)**.** *When $T = m - 1 + (k+1)\bar{d}$ for some $k \in \mathbb{N}_0$ then the diametric bound holds. Otherwise as $T \to \infty$ then the diametric bound holds.*

Proof: **??**

Fixing the diametric attack

A possible "fix" to the problem of the excess time is to limit the diametric attacks window in which attacks are placed.

Definition 2.3 (Time-limited diametric attack)**.** Attacking at a pair of diametric nodes equiprobably for the times I,$I+1$...,$I+\bar{d}-1$ (i.e starting attacks at $\tau, \tau+1, ..., \tau+\bar{d}-1$) is called the *timed diametric attack.*

Note. The time-limited diametric attack is only feasible is $T \geq m + \bar{d} - 1$.

Lemma 2.4. *When $T \geq m + \bar{d} - 1$, the diametric bound $V \leq \max\{\frac{1}{2}, \frac{m}{2\bar{d}}\}$ is valid.*

Proof: **??**

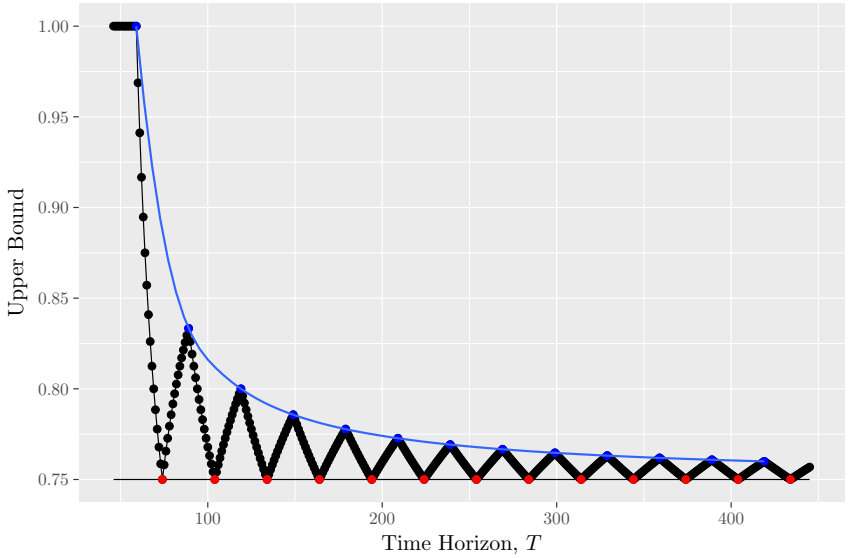
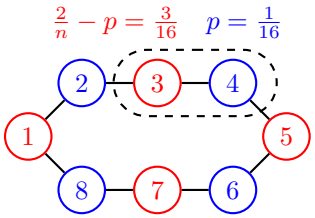


Figure 1: Best Upper Bound achievable under the diametric strategy

2.2 New Tools

We seek to find a larger class of optimal patrols for hamiltonian graphs, by using groups of nodes.

Definition 2.5 (Alternating Random Hamiltonian Patrol(ARHP))**.** An *Alternating Random Hamiltonian Patrol* (*ARHP*) is a mixed strategy following the Hamiltonian cycle but with a probability p of starting at "even" nodes and a probability of $\frac{2}{n} - p$ of starting at "odd" nodes.



Example Figure 2.1: C_8 with the blue nodes being "even" nodes started at with probability $\frac{1}{16}$ and the red nodes being "odd" nodes started at with probability $\frac{3}{16}$.

Lemma 2.6. *When n and m are both even, following the Alternating Random Hamiltonian Patrol, if feasible, gives the same lower bound as the random Hamiltonian patrol, i.e $V \geq \frac{m}{n}$.*

Proof. During any attack interval I which is of even length, then $W(I)$ contains m' "even" and m' "odd" nodes for a total of $m = 2m'$ nodes. Therefore by

following the Alternating Random Hamiltonian Patrol, $\boldsymbol{\pi}_{ARHP}$, with probability p at "even" nodes and probability $\frac{2}{n} - p$ at "odd" nodes. Then

$$P(\boldsymbol{\pi}_{ARHP}, [i, I]) \geq \underbrace{\overbrace{p}^{\text{even node}} + \overbrace{\frac{2}{n} - p}^{\text{odd node}} + p + \frac{2}{n} - p + ... + p + \frac{2}{n} - p}_{m=2m' \text{ elements}}$$

$$= m'p + m'(\frac{2}{n} - p) = \frac{2m'}{n} = \frac{m}{n} \quad \forall i \in N \quad \forall I \subseteq \mathcal{T}$$

Hence as it holds for all pure attacks

$$P(\boldsymbol{\pi}_{ARHP}, \boldsymbol{\phi}) \geq \frac{m}{n} \quad \forall \boldsymbol{\phi} \in \Phi$$

Hence $V \geq \frac{m}{n}$. □

If m is odd, say $m = 2m' + 1$ then in the above we get two possibilities for each node depending on the interval choice either $p + \frac{m-1}{n}$ or $\frac{m+1}{n} - p$. So choosing anything other than $p = \frac{1}{n}$ (which is the Random Hamiltonian Patrol strategy) gives a worse result for the patroller.

While not getting a better lower bound, the ARHP does give some control on how to perform optimally in a Hamiltonian graph. The idea of distributing the probability $\frac{2}{n}$ between two types of nodes can be extended to the idea of distributing the probability $\frac{k}{n}$ between k types of nodes (as seen in appendix **??**).

2.3 Star graph solution

As a special case of Complete bipartite we have the star graph, $S_n = K_{1,n}$, that is a tree with one internal node (the centre) and n leaf nodes (the external nodes). Hence $V(S_n) = V(K_{1,n}) = \frac{m}{2n}$ for $m < 2n$ (by the Complete bipartite corollary(1.18)). This is achieved by the patroller forming a patrol which alternates between different external nodes and the centre, and the attacker attacking at all the external nodes with equal probability (for a fixed time interval).

2.4 Extending the star graph

2.5 Joining star graphs by centralised connections

3 Strategic Patroller with Random Attackers

In this section we provide further work developed from the literature review in Section 1.3.

3.1 Deterministic Attack time experiments

As the theory works for all bounded attack time distributions, we shall look at the deterministic attack time, using $P(X_j = x_j) = 1$, we can note that this does some level of reduction on the state space as costs are only incurred at states with $s_j = B_j$ or $B_j + 1$. The cost function reduces to

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = \begin{cases} c_j\lambda_j R_j + c_j v_j \text{ for } s_j = B_j, i \neq j \\ c_j\lambda_j \text{ for } s_j = B_j + 1, i \neq j \\ 0 \text{ Otherwise} \end{cases}$$

Looking at the single node problem, we notice that there is no cost to transition to the state $s = B$, so at this point we must decide if we are renew now or renew at $s = B + 1$ or never renew at all (If indifferent between $B + 1, B + 2, ...$)

It is curious whether this sudden spike in the cost (and therefore in the resulting index) will cause any issue with the proposed heuristics.

This numerical experiment needs to be done

4 Strategic Patroller with Random Attackers and Local-observations

We now look at altering the work summarized in Section 1.3 to incorporate suspicious behaviour of attackers who arrive while the patroller is present.

4.1 Altering the problem for instantaneous moving patroller

As the standard set-up has a patroller moving at unit speed along the edges, they are never really present at a node (even when waiting). As we wish to have the patroller observe some arrivals we shall first alter this assumption to that of an instantaneous moving patrolling. For now we will use the same assumption that attackers who arrive while the patroller is present are caught, however we will soon assume they are smart enough to not start their attack and instantly get caught.

While the standard set-up remains mostly intact, the cost function is slightly altered to accommodate this change. We note there is now a difference in cost depending on if the node is visited or not.

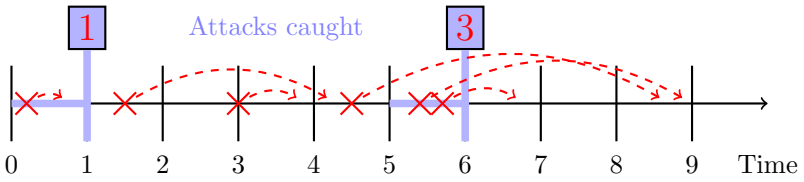


Figure 4.1: Example of timing

$$C_j(\boldsymbol{s}, i) = \begin{cases} 0 \text{ if } j = i \\ c_j \lambda_j \int_{s_j - 1}^{s_j} P(X_j \leq t) dt \text{ if } j \neq i \end{cases}$$

This change does not the theory that much, the problem can still be reduced to the single node version of the problem. However there is a slight change here, which results in a slight change to the optimal solution and hence the index which is suggested.

The following are the changes

$$f(k) \equiv \frac{c\lambda \int_0^{k-1} P(X \leq t) dt + \omega}{k}$$
$$W(k) \equiv c\lambda \left(\int_{k-1}^{k} P(X \leq t) dt - \int_0^{k-1} P(X \leq t) dt \right)$$

Note. $W(0) = 0$ and for $k \geq B + 1$ $W(k) = c\lambda E[X]$

Using this new definition of $W(k)$, Theorem 1.19 still holds and gives us the optimal policy. Hence suggesting an index of

$$W_i(\boldsymbol{s}) \equiv c_i \lambda_i \left(\int_{s_i - 1}^{s_i} P(X_i \leq t) dt - \int_0^{s_i - 1} P(X_i \leq t) dt \right)$$

For completness sake, we provide numerical results for the IRH and IPH defined as before, but using this index.

4.2 Numerical results for instantaneously moving patroller

For completeness sake, we provide numerical results for the IRH and IPH defined as before, but using this index.

THIS NEEDS TO BE DONE

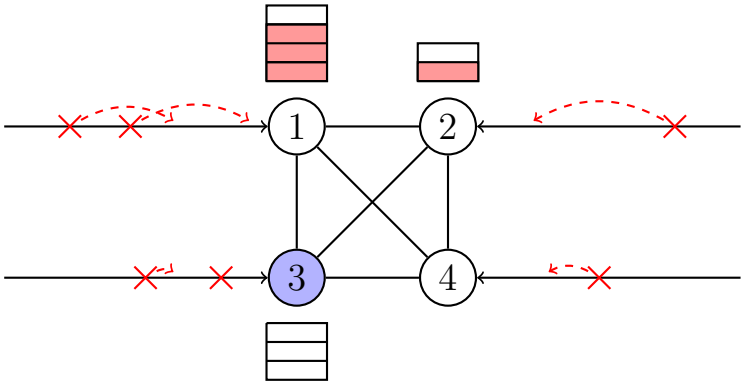


Figure 2: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

4.3 Altering problem to accommodate local-observable information

Now that we have established a instantaneously moving patroller, we now introduce the idea of local-observations, that is the patroller witnesses attackers arriving at there current node, but the attackers do not begin there attack immediately,as they are aware of the patroller's presence. For now we assume that any such attacker is only able to delay their attack to start at the beginning of the next time period.

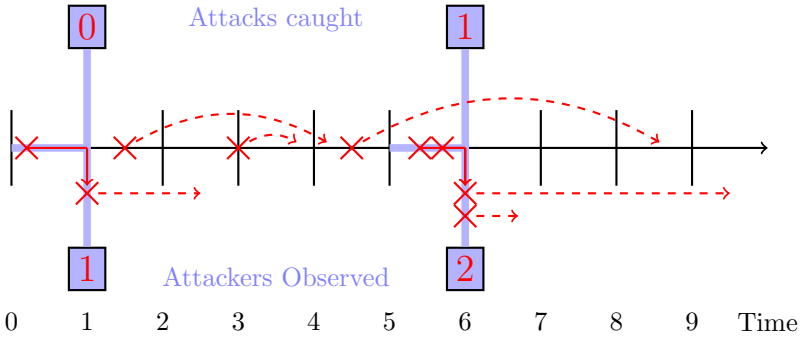


Figure 4.2: Example of timing

The problem can still be formulated as a MDP, however our states now need to hold the information about how many attackers where observed when the node was last visited. Our state space is

$$\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) = (s_1, ..., s_n, v_1, ..., v_n) \,|\, s_i = 1, 2, ..., \ v_i = 0, 1, ... \text{ for } i = 1, ..., n\}$$

Where as before, s_i denotes the time since the last decision to visit node i was

taken and the newly introduced v_i denotes how many attackers where observed when the patroller last visited node i.The current node is $l(\boldsymbol{s}, \boldsymbol{v}) = \text{argmin}_i\, s_i$ and the decisions from $(\boldsymbol{s}, \boldsymbol{v})$ are still adjacent node, $\mathcal{A}(\boldsymbol{s}, \boldsymbol{v})$. The transition when node i is chosen to move to are $\phi(\boldsymbol{s}, \boldsymbol{v}, i) = (\widetilde{\boldsymbol{s}}, \widetilde{\boldsymbol{v}})$, where; $\widetilde{s}_i = 1$, $\widetilde{s}_j = s_j + 1\ \forall j \neq i$, $\widetilde{v}_i \sim Po(\lambda_i)$ and $\widetilde{v}_j = v_j\ \forall j \neq i$.

That is the $\boldsymbol{s}$ state transitions as before, and the $\boldsymbol{v}$ state updates the chosen node to be the amount observed while the patroller is at the node, which is drawn from a Poisson distribution of rate λ_i due to the arrivals being a Poisson Process.

Again the future of the process is independent of its past, so we can formulate its movement as an MC and hence the patrollers problem is a MDP.

The patroller incurs cost c_j for all successful attacks at node j. A successful attack will fall into two categories; an observed attack or an attacker who arrived. We simply sum the expected costs of both types of attacker to work out the cost at a node for the next unit time, given we choose to move to node i.

$$C_j(\boldsymbol{s}, i) = \begin{cases} 0 \text{ if } j = i \\ \underbrace{c_j P(s_j - 1 < X_j \leq s_j)}_{\text{Observed finishing}} + \underbrace{c_j \lambda_j \int_{s_j - 1}^{s_j} P(X_j \leq t) dt}_{\text{Arrivals finishing}} \text{ if } j \neq i \end{cases}$$

We run into the same problem as before, with a countably infinite state space, Ω, we therefore wish to bound ourselves to a finite state space. As before we can bound the attack times, defining B_j as in 1, but this only partially solves our problem, as it bounds the $\boldsymbol{s}$ part of the state space but not the $\boldsymbol{v}$ part.

To bound $\boldsymbol{v}$ we introduce a observable capacity, $\boldsymbol{b}$, where b_i is the maximum number of local-observations at node i. We assume that all other attackers fail once this capacity is reached.

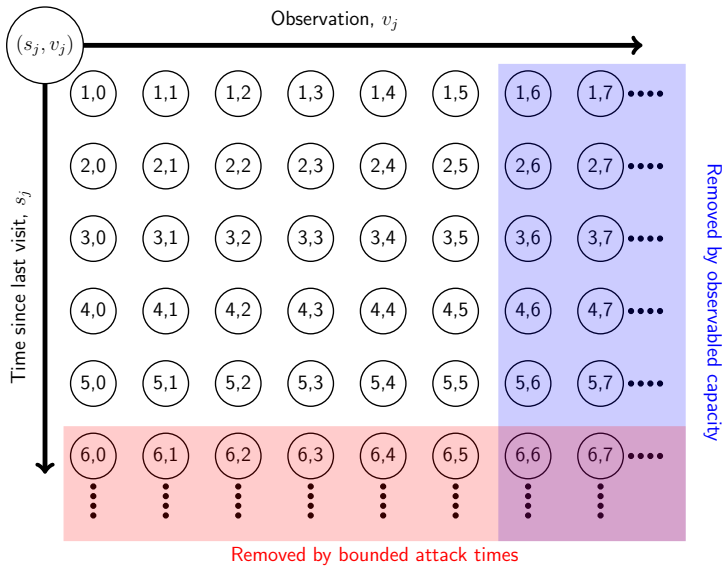


Figure 3: State space diagram for node j, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

This change to a finite state space, $\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) = (s_1, ..., s_n, v_1, ..., v_n) \,|\, s_i = 1, 2, ..., B_i + 1 \;\; v_i = 0, 1, ..., b_i \text{ for } i = 1, ..., n\}$ and caps the transitions. That is that $\widetilde{s}_j = \min\{s_j + 1, B_j + 1\}$ and $\widetilde{v}_i \sim TPo(\lambda_i, b_i)$, where $TPo(\lambda, b)$ is the truncated Poisson distribution, with all the tail probability at the value b_i. I.e.

$$P(TPo(\lambda, b) = \begin{cases} P(Po(\lambda) \text{ if } i \neq b \\ P(Po(\lambda) \geq i) \text{ if } i = b \\ 0 \text{ Otherwise} \end{cases}$$

Even though the state space is now finite, the transitions are not deterministic, so a cyclic behaviour is not induced when the same state is reached again. For notational purposes we will still use $\phi_\pi^k(\boldsymbol{s}_0, \boldsymbol{v}_0)$ to be the state after k transitions from $(\boldsymbol{s}_0, \boldsymbol{v}_0)$.

More work is needed to see if the cyclic behaviour property does still hold just in some probabilistic fashion

This problem, as before, can be solved by standard techniques such as value iteration or linear programing to compute the long-run average cost. This is left to Appendix C.

We can then use the standard reduction tools of making the problem into a MN problem and then applying the TR constraint and/or the Lagrangian relaxation. We shall therefore focus on the Lagrangian relaxation which is equivalent to the single node problem.

That is we wish to minimize $V(\pi)+\omega\mu(\pi)$. Our state space is more complicated than before due to the inclusion of the time since it was last visited and the amount of local-observations made when it was last visited.

4.4 Deterministic attack time assumption

Because our state space is hard to deal with,in the single node problem, we shall reduce the problem, by assuming that the attack times are deterministic, that is $P(X = x) = 1$, this reduces our cost function to

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = \begin{cases} c_j\lambda_j R_j + c_j v_j \text{ for } s_j = B_j, i \neq j \\ c_j\lambda_j \text{ for } s_j = B_j + 1, i \neq j \\ 0 \text{ Otherwise} \end{cases}$$

Where $R_j = B_j - x_j$. We see that we only worry about incurring costs in states when $s_i = B_i$ or $B_i + 1$ for some i.

This allows us to see a clear solution to the single node problem.

4.5 Single node solution

Again we wish to minimize the long-run average cost, paying ω when we choose to visit a node. We note as mentioned in Section [refer to section on not cycling] that we are not in a cycle, so will take a different approach. At each state our choice is binary; we either wait or renew/visit.

We wish to solve

$$g = \min_{\pi_1 \in \Pi_1^{\text{MN}}} V(\pi_1) + \omega\mu(\pi_1)$$

And because g is the long-run average cost, we know that in the limit of $n \to \infty$ that $V_n(s, v) = ng + h(s, v)$ where $h(s, v)$ is bias from starting at the state (s, v) rather than in equilibrium.

It should be clear that as the cost of all states where $s < B$ are zero, we should wait until $s = B$ before making any form of decision. However for completeness sake we present a formal argument

From any (s, v) with $s < B$ consider the policy π_k which waits k time periods and then renews and follows some optimal policy, σ, with $k = 0, ..., B - s$.

Using such a policy will get us that

$$V_n^{\pi_k}(x, v) = \omega + E[V_{n-k-1}^{\sigma}(\theta)] \tag{2}$$

where θ is the state upon renewal (i.e it is the state $(1, V) \sim (1, TPo(\lambda, b))$.

Now we will pick policy π_{k+1} over π_k (or be indifferent) if

$$\begin{aligned} &\lim_{n\to\infty} V_n^{\pi_k}(x, v) - V_n^{\pi_{k+1}}(x, v) \geq 0 \\ \iff &\lim_{n\to\infty} E[V_{n-k}^{\sigma}(\theta) - V_{n-k-1}^{\sigma}(\theta)] \geq 0 \\ \iff &\ g \geq 0 \end{aligned}$$

As we only have positive costs, we know that $g \geq 0$ and hence the above argument shows that in such a state it is always best to wait another time period and then make a decision, making the same decision to wait again if we have not reached $s = B$.

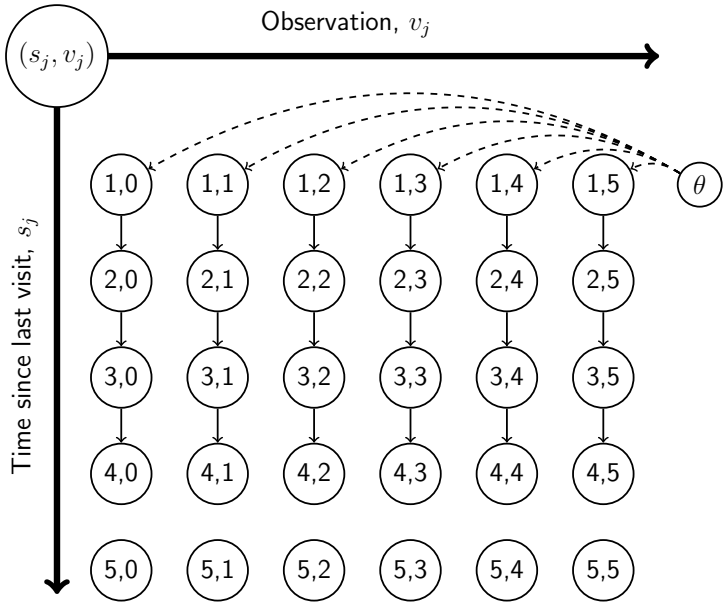


Figure 4.3: Optimal movements for $s < B$

We will look at getting a similar idea for the states $s = B + 1$, as it should be clear that if we do not renew initially then as our state does not change, we will never renew. However for completeness sake we will provide a formal argument.

From any $(B+1, v)$ consider the policy π_k which waits k time periods and then renews and follows some optimal policy, σ, with $k = 0, ...$

Using such a policy will get us that

$$V_n^{\pi_k}(s, v) = kc\lambda + \omega + E[V_{n-k-1}^{\sigma}(\theta)] \tag{3}$$

Again we will pick a policy π_{k+1} over π_k (or be indifferent) if

$$\begin{aligned}
&\lim_{n\to\infty} V_n^{\pi_k}(\lfloor B \rfloor + 2, 0) - V_n^{\pi_{k+1}}(\lfloor B \rfloor + 2, 0) \geq 0 \\
&\iff \lim_{n\to\infty} -c\lambda + E[V_{n-k}^{\sigma}(\theta) - V_{n-k-1}^{\sigma}(\theta)] \geq 0 \\
&\iff g \geq c\lambda
\end{aligned}$$

Now we introduce a policy, $\pi_{\text{Neg}}(s, v) = 0 \ \forall (s, v) \in \Omega$, the policy which never renews. It is clear that this policy gives a long-run average cost of $c\lambda$, so $g \leq c\lambda$ and hence we will always choose to renew immediately in $s = B + 1$.

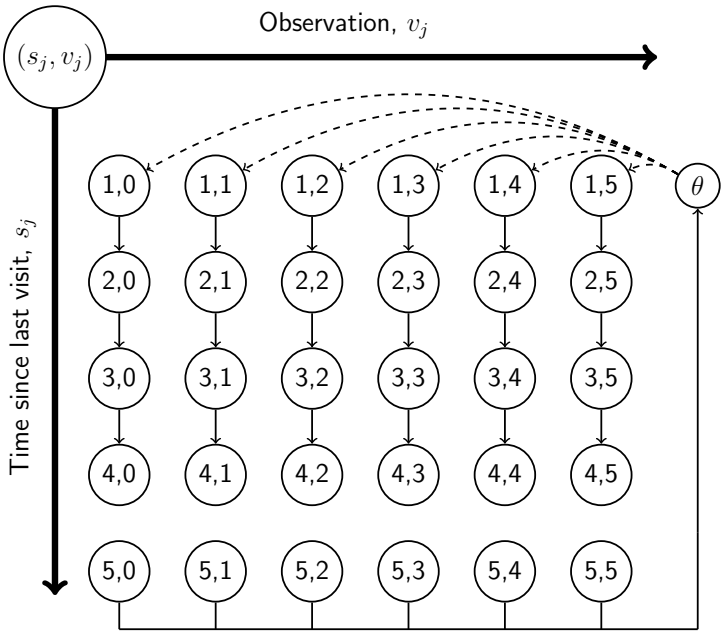


Figure 4.4: Optimal movements for $s < B$ and $s = B + 1$

Because either side of our states with $s = B$ we know the decision, we can see in states (B, v) whether we should renew now or wait one period and then renew. Using a similar argument to the two previously, we shall consider a policy, π^{R} a policy which renews immediately and a policy π^{NR} which does not renew immediately, but renews in one time period, which both after these decisions follow some optimal policy, σ from the renewed state, θ. Using Such policies will get us

$$V_n^{\pi_{\mathrm{R}}}(s,v) = \omega + E[V_{n-1}^{\sigma}(\theta)]// \quad V_n^{\pi_{\mathrm{NR}}}(s,v) = c(\lambda R + v)\omega + E[V_{n-1}^{\sigma}(\theta)]//$$

We will pick policy, π^{R} over π^{NR}

$$\begin{aligned}
&\lim_{n\to\infty} V_n^{\pi_{NR}}(B,v) - V_n^{\pi_R}(B,v) \geq 0 \\
&\iff \lim_{n\to\infty} c(\lambda R + v) + E[V_{n-1}^{\sigma}(B+1,0)] - (\omega + E[V_{n-1}^{\sigma}(\theta)]) \geq 0 \\
&\iff \lim_{n\to\infty} c(\lambda R + v) + E[\omega + V_{n-2}^{\sigma}(\theta)] - \omega - E[V_{n-1}^{\sigma}(\theta)] \geq 0 \\
&\iff \lim_{n\to\infty} c(\lambda R + v) + E[V_{n-2}^{\sigma}(\theta) - V_{n-1}^{\sigma}(\theta)] \geq 0 \\
&\iff c(\lambda R + v) - g \geq 0 \\
&\iff g \leq c(\lambda R + v)
\end{aligned}$$

Meaning that in state, $s = B$ we are dependent on v and if we have that $g \leq c(\lambda R + v)$ then we will renew immediately. We see that if we renew now in v we will definitely renew in $v+1$ (as $g \leq c(\lambda R + v) \implies g \leq c(\lambda R + v + 1)$).This motivates the definition of the type of optimal policy, depending on some threshold.

Definition 4.1 (Threshold policy)**.** A policy, $\pi_{\mathrm{Th}}(v_{\mathrm{crit}})$, is said to be a *threshold* policy, with threshold, v_{crit}, if:

- In states (s,v) , $s < B, \forall v$ it waits until (B,v)
- In states (B,v) it renews now if $v \geq v_{\mathrm{crit}}$ and waits until $(B+1,v)$ if $v < v_{\mathrm{crit}}$
- In states $(B+1,v)$ $\forall v$ it renews now.

But due to knowing that, $g \leq c\lambda$, we may have some maximum threshold. We will define

$$v_{\max} \equiv \max\{v \in \{0,1,...,b\} \mid v \leq \lambda(1-R)\}$$

Then we may have $v_{\mathrm{crit}} \in \{0,1,...,v_{\max}+1\}$.

While we do not yet know, g, we can consider splitting it up into regions which imply using different threshold policies.

- $g \leq c\lambda R$ gives $v_{\mathrm{crit}} = 0$.
- $c\lambda R + c(k-1) < g \leq c\lambda R + kc$ gives $v_{\mathrm{crit}} \neq 0, v_{\max}+1$
- $c\lambda R + cv_{\max} < g \leq c\lambda$ gives $v_{\mathrm{crit}} = v_{\max}+1$

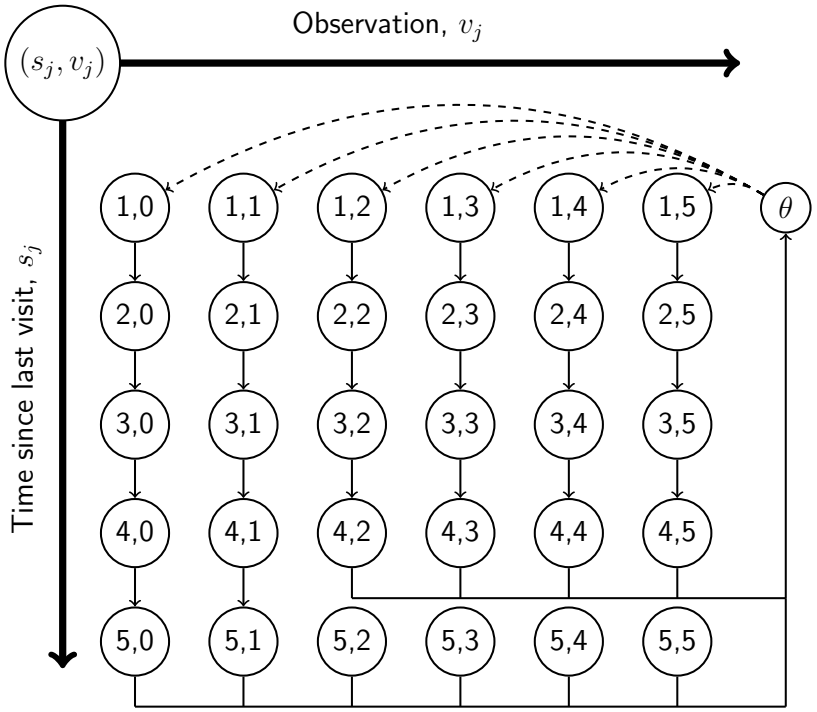


Figure 4.5: Threshold policy, $\pi_{\text{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Given that we are using a threshold policy, with threshold π_{crit}, we can work out the long-run average cost as a function of ω

$$g^{\pi_{\text{Th}}(v_{\text{crit}})}(\omega) = \frac{\text{Expected cost per renewal}}{\text{Expected renewal length}}$$
$$= \frac{\omega + c\lambda R(1 - P(TPo(\lambda, b) \geq v_{\text{crit}})) + c \sum_{i=0}^{v_{\text{crit}}-1} iP(TPo(\lambda, b) = i)}{B + 1 - P(TPo(\lambda, b) \geq v_{\text{crit}})}$$

This allows us to convert our previous bounds on, g, which imply a certain threshold should be picked, to bounds on our visitation cost, ω.

- $0 \leq \omega \leq c\lambda R(B+1) \equiv \Delta(0)$ if $v_{\text{crit}} = 0$.
- $\Delta(v_{\text{crit}} - 1) < \omega \leq \Delta(v_{\text{crit}})$ if $v_{\text{crit}} \neq 0, v_{\max}$
- $\Delta(v_{\max}) < \omega \leq \widetilde{\Delta}$ if $v_{\text{crit}} = v_{\max} + 1$

Where

- $\Delta(k) = c(\lambda RB + k(B + 1 - P(TPo(\lambda, b) \geq k)) - \sum_{i=0}^{k-1} iP(TPo(\lambda, b) = i))$

- $\widetilde{\Delta} = c(\lambda(B+1-R+(R-1)P(TPo(\lambda,b) \geq v_{\max}+1)) - \sum_{i=0}^{v_{\max}} iP(TPo(\lambda,b) = i))$

And we note, that $\widetilde{\Delta}$ is the 'capped' version of Δ due to the bound of $g \leq c\lambda$.

Hence we have solved the single node problem, and hence $C(\omega)$.

4.6 Index and heuristics

As we know the strategy at a single node, we can imagine in all states 'bidding' for how much they want to be visited from the current state of the system, 'bidding' a 'fair price'.

This fair price can be seen from [reference to omega equations], and considering what a node would be willing to 'bid' to be visited now.

Consider a state (s, v) with $s < B$ then the node is not willing to pay for a visit (or will pay zero), so these states are given an index of zero.

To find the fair cost in states (B, v) consider that they will be renewed in threshold policy with $v_{\text{crit}} = 0, 1, ..., v$ policy, but not for any higher thresholds, so the maximum service price it is willing to pay is $Delta(v)$ for visit now.

Note. Due to $v_{\max}$ this can be capped to $\widetilde{\Delta}$ for $v \geq v_{\max} + 1$

To find the fair cost in $(B + 1, v)$ consider that for all thresholds we are willing to renew in these states and are willing to pay up to $\widetilde{\Delta}$.

This gives us a function which can help to classify the optimal solution for the single-node problem.

$$W(s,v) = \begin{cases} 0 \text{ If } s < B\,, \\ \Delta(v) \text{ If } s_i = B_i, v_i \leq v_{\max}\,, \\ \widetilde{\Delta} \text{ If } s = B, v \geq v_{\max} + 1\,, \\ \widetilde{\Delta} \text{ If } s = B + 1\,. \end{cases}$$

Then if $\omega \leq W(s, v)$ the optimal action is to renew now.

We can reinsert the node subscript, i, and attempt to use this index to aim to implement it as the price a node is willing to pay to be visited

$$W_i(\boldsymbol{s},\boldsymbol{v}) = \begin{cases} 0 \text{ If } s_i < B_i\,, \\ \Delta_i(v_i) \text{ If } s_i = B_i, v_i \leq v_{i,\max}\,, \\ \widetilde{\Delta}_i \text{ If } s_i = B_i, v_i \geq v_{i,\max}+1\,, \\ \widetilde{\Delta}_i \text{ If } s_i = B_i+1\,. \end{cases}$$

Now we can form a simple heuristic, called the Index Heuristic(IH), which just picks to go an adjacent node with the highest index.

We can see the index as a benefit for visiting the selected node. The patroller can use a look-ahead window of length, l, that is to look at all paths of length l and sum all the benefits (indices) collected along the path and using the maximum path for our immediate action. We call this heuristic the Index Benefit Heuristic(IBH).

We can repeat this for every state to get a look-up table for the action to take in any given state.

We notice that as we are looking at summing all indices for all l steps in the look ahead window, we have already calculated this for all look ahead windows $l' \leq l$. We might as well use this information, as it is being computed. We shall call this the heuristics depth, d, and denote IBH(d) to be the index benefit heuristic that look at all look ahead windows $l = 1, ..., d$ and for each l returns an action to take.

To select between the d paths (and therefore actions) from the different look ahead windows, we need some form of path selection heuristic. We decide to use a proxy for the long-run average cost, by analysing the expected short term average cost along the path and biasing this with the average cost to decay to the fully neglected state $(\boldsymbol{B}+\mathbf{1},\boldsymbol{v})$ from the end of the path's state by the patroller leaving the system (and no longer visiting any node).

Instead of seeing the index as a benefit for selecting a node, we can see the index as a penalty for not selecting a node. The patroller can use a look-ahead window of length, l, and sum all the penalties (indices for nodes not selected) collected along the path and using the minimum path for our immediate action. We call this heuristic the Index Penalty Heuristic(IPH) and similarly we can look at depth d called the IPH(d) by comparing all look-ahead window choices of lengths $l = 1, ..., d$ and using the same path selection as IBH(d).

Note. IBH(1), IPH(1) are the same heuristic

We could also consider a short-sighted for numerical study purposes. We know in state $(\boldsymbol{s},\boldsymbol{v})$ and the patroller choosing node i then the expected number of attacks they can detect is $\lambda_i \int_0^{s_i-1} P(X_i > t)dt + v_i P(X_i > s_i - 1)$. Therefore we define the cost avoided (reward gained) if the patroller visits i from state $(\boldsymbol{s},\boldsymbol{v})$

$$R(\boldsymbol{s}, \boldsymbol{v}, i) = c_i(\lambda_i \int_0^{s_i - 1} P(X_i > t)dt + v_i P(X_i > s_i - 1)$$

Using this reward function for selecting a node is an option instead of seeing the index as some benefit is an option.The patroller can use a look-ahead window of length, l, and sum all the rewards collected along the path and using the maximum path for our immediate action. We call this heuristic the Myopic Heuristic(MH) and similarly we can look at depth d called the MH(d) by comparing all look-ahead window choices of lengths $l = 1, ..., d$ and using the same path selection as IBH(d).

Note. Due to the path selection method used for the depth heuristics, we cannot guarantee that increasing the depth increases the overall optimal answer provided by the heuristics.

4.7 Numerical experiments

We now Look at using our heuristic, at certain depths to see how good it is against optimal. To do this we generate 500 scenario's on the complete graph, K_4 with attack times, X_i, on the 4 nodes picked uniformly from $[1, 3]$, we lock $b_i = 1 \ \forall i$ and allow each arrival rates, λ_i, to be picked uniformly from $[0, 1]$ and use $c_i = 1 \ \forall i$ so that the cost incurred can be interpreted as the probability of missing an attack.

For the same scenario, we run BH(d),PH(d) for $d = 1, 2, 3, 4$ and return the percentage error.

From Figure **??** we can see that while sometimes the heuristic performs well, in the majority of cases it performs very badly.

We propose altering the index to

$$W_i(s_i, v_i) = \begin{cases} 0 \text{ If } s_i < B_i \\ \Delta_i(v_i) \text{ If } s_i = B_i \\ \Delta_i(v_i + 1) \text{ If } s_i = B_i + 1, v_i < v_{i,\max} \\ \widetilde{\Delta_i} \text{ If } s_i = B_i + 1, v_i \geq v_{i,\max} \end{cases} \tag{4}$$

We run the same scenarios with our changed index

As we can see from Figure **??** the change to index, significantly. We can also note that on the complete the penalty heuristic seems to outperform the benefit heuristic. This agrees with the findings in Lin et al. [2013].

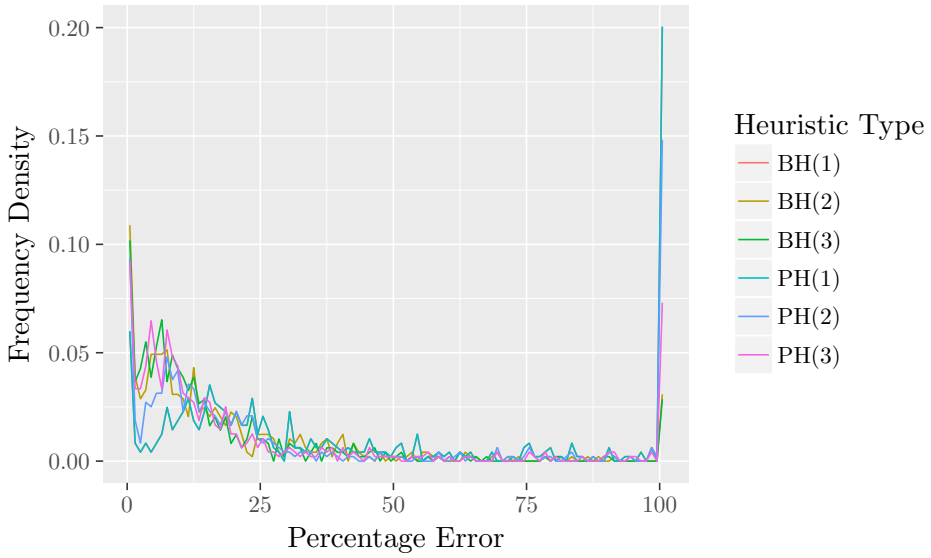


Figure 4.6: Frequency density of percentage errors made by heuristics in simulations

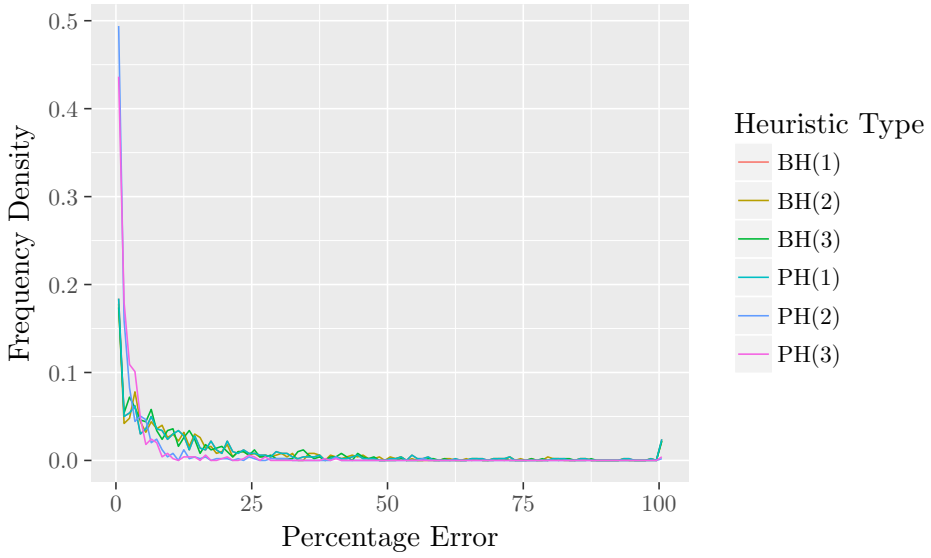


Figure 4.7: Frequency density of percentage errors made by heuristics in simulations for new index

5 Future Work

5.1 Proof completions

5.2 Extending Discrete Attack time to generic distributions

We want to look at removing the restriction placed on the attack time to be deterministic, as in section 4.4. This will allow us to deal with any generic distribution, to do so we will focus on the single node problem with a generic distribution.

As before if we start in states $(B+1, v)$ we will renew if $g \leq c\lambda$ and by the neglecting strategy we know this to guaranteed. So the optimal strategy will have all states with $s = B+1$ renew immediately.

No we will work backwards, aiming to use backwards recursion to work through to some multi-stage threshold policy, $\pi^{\mathrm{TH}}(\boldsymbol{v}_{\mathrm{crit}})$ where $(\boldsymbol{v}_{\mathrm{crit}})_i$ is the threshold on the state space row of $s = i$. We expect this threshold to be triangular on the state space, that is that $\boldsymbol{v}_{\mathrm{crit}}$ is a non-decreasing vector.

Definition 5.1 (Multi-stage threshold policy)**.**

As an example we will now start to work out what to do in states $s = B$ and how to work out the threshold. Well the same process as in Section 4.5 holds and we will renew if $g \leq c(\lambda R + v)$ otherwise we will wait and renew later.

Now to decide if in $(B-1, v)$ we should renew or not we need to consider waiting zero,one or two periods before renewing (renew in $s = B-1, B, B+1$).

We could split ourself into cases, of v such that $g \leq c(\lambda R + v)$ (so if we wait we renew in $s = B$) or otherwise we wait till $s = B+1$.

5.2.1 Plan

Time Estimate: 3 Months

Plan:

5.3 Strategic Patroller with Random Attackers on Edges

We want to look at considering similar work to that in section, 1.3 but instead of attackers arriving at nodes, we could consider attackers arriving on edges. We provide a brief introduction to this idea now focusing on directed graphs.

We use an directed graph, $Q = (N, E)$, with nodes $N = \{1, ..., n\}$ and edges

$E =\subset E_{\text{Comp}} = \{(i,j)\,|(i,j) \in N\}$. We have our adjacency matrix, A, where $A_{i,j}$ means that a transition from node i to node j is possible.

Attackers arrive at each edge according to a Poisson process of rate, $\lambda_{i,j}$ and pick a position along the unit length edge according to a distribution function, $f_{i,j}(x)$ (with support, $x \in [0,1]$, and c.d.f, $F_{i,j}(x)$). The attack lasts, $X_{i,j}$ time units before completion and a cost is incurred to the patroller of $c_{i,j}$.

The patrollers strategy is some walk (with possible waiting), we will assume as before that they walk along the edges at unit speed. The decision are which edge to traverse (which is equivalent to which node to walk to).

Our state space is $\Omega = \{\boldsymbol{s} = (s_{i,j})_{(i,j)\in E}\,|\,s_{i,j} = 1, 2, ...\ \forall (i,j) \in E\}$, where $s_{i,j}$ is the number of time periods since the edge (i,j) was last traversed.

From a state, $\boldsymbol{s}$, we can identify the current node by $l(\boldsymbol{s}) = (\text{argmin}_{(i,j)}\,\boldsymbol{s})_2$ the patroller can choose to move to $\mathcal{A}(\boldsymbol{s}) = \{j'\,|(A)_{l(\boldsymbol{s}),j'} = 1\}$ and choosing node, j',is equivalent to choosing to traverse edge (j,j'). When node, j' is chosen to be moved to we transition to state, $\phi(\boldsymbol{s},j') = \widetilde{\boldsymbol{s}}$ where $s_{l(\boldsymbol{s}),j} = 1$ and $s_{i,j} = s_{i,j} + 1\ \forall (i,j) \in E$
$\{(l(\boldsymbol{s}),j')\}$.

Again the future of the process is independent of its past, we can formulate its movement as a MC and the patroller's problem is a MDP.

The patroller incurs a cost at each edge for the next time period I.e $C(\boldsymbol{s},j') = \sum_{(i,j)\in E} C_{i,j}(\boldsymbol{s},j')$ where $C_{i,j}(\boldsymbol{s},j')$ is the cost at the edge (i,j) choosing to move to node j' in the next time period. Hence

$$\begin{aligned} C_{i,j}(\boldsymbol{s}) &= c_{i,j}\lambda_{i,j}\int_0^1 f_{i,j}(y)\int_0^{s_{i,j}} P(t-1 < X_{i,j} \le t)dtdy \\ &= c_{i,j}\lambda_{i,j}\int_0^{s_{i,j}} P(t-1 < X_{i,j} \le t)dt \end{aligned}$$

Note. We note due to all edges being traversed at the same speed, the last time a point x (from node i) along the edge (i,j) was seen was $s_{i,j} - x$, so when it returned to it is seen in x time, so is seen again in $s_{i,j}$ time.

As before we bound the attack times by, using $B_{i,j} \equiv \min\{k\,|k \in \mathbb{Z}^+ P(X_{i,j} \le k) = 1\}$ to create a finite state space, $\Omega = \{\boldsymbol{s}\,|s_{i,j} = 1, 2, ..., B_{i,j}\ \forall (i,j) \in E\}$ and our transition is modified to $\widetilde{s}_{i,j} = \min\{s_{i,j}+1, B_{i,j}+1\}\ \forall (i,j) \in E$
$\{(l(\boldsymbol{s}),j')\}$.

The objective again is to minimize the long-run average cost among all edges and as we have a finite state space as before, we can focus on the class of stationary, deterministic policies, $\Pi = \{\pi : \Omega \to E\,|\pi(\boldsymbol{s}) \in \{(l(\boldsymbol{s}),j')\,|\,j \in \mathcal{A}(\boldsymbol{s})\}\}$ and we wish to solve

$$C^{\mathrm{OPT}}(\boldsymbol{s}_0) \equiv \min_{\pi \in \Pi} \sum_{(i,j) \in E} V_{i,j}(\pi, \boldsymbol{s}_0)$$

Where $V_{i,j}(\pi, \boldsymbol{s}_0)$ is the long-run average cost incurred at edge (i, j) under the policy, π, starting from state, $\boldsymbol{s}_0$, defined by,

$$V_{i,j}(\pi, \boldsymbol{s}_0) \equiv \lim_{N \to \infty} \frac{1}{N} \sum_{(i,j) \in E} C_{i,j}(\phi_\pi^k(\boldsymbol{s}_0), \pi(\phi_\pi^k(\boldsymbol{s}_0)))$$

Where $\phi_\pi^k(\boldsymbol{s}_0)$ is the state after k transitions starting from $\boldsymbol{s}_0$ under the policy π.

We now notice and extreme similarity to the 'solved' problem in 1.3 and in-fact we can see a link between the edges here and the nodes in the original proble. To seek some correspondence between the two problems by using the edge-vertex dual for a directed graphs (as in).

Note. The resultant edge-vertex dual graph is directed, but the original problem has no issue with this.

Do I need to show this 1-1 correspondence ?

This correspondence, allows us to use the heuristics and theory as in 1.3.

The idea now is does this still work with undirected graphs, we shall note that the cost function changes slightly and the way we traverse the graph matters due to the orientation of the arrivals at positions.

5.3.1 Plan

Time Estimate: 2 Months

Plan: To extend the theory to deal with the undirected graph, which will include modifying the cost function and state space for the opposite traversal's . Then to isolate particular examples which can be 'solved' with the previous reduction tools; such as acyclic graphs, which remove the dependency on which way the edges are traversed. The study of the class of acyclic graphs will be important as all walks which consider using the same edge always traverse it in the opposite direction to as before. Then to see if there is a way to convert an undirected problem to a directed problem and hence use the correspondence for the solution. Finally we wish to try to solve it from first principals without any intuitive understanding.

References

Steve Alpern, Alec Morton, and Katerina Papadaki. Patrolling Games. *Operations Research*, 59(5):1246–1257, 2011. doi: 10.1287/opre.1110.0983. URL `http://pubsonline.informs.orghttp://`.

Katerina Papadaki, Steve Alpern, Thomas Lidbetter, and Alec Morton. Patrolling a Border. *Operations Research*, 64(6):1256–1269, 2016. doi: 10.1287/opre.2016.1511. URL `http://pubsonline.informs.orghttp://dx.doi.org/10.1287/opre.2016.1511`.

Kyle Y Lin, Michael P Atkinson, Timothy H Chung, and Kevin D Glazebrook. A Graph Patrol Problem with Random Attack Times. *Operations Research*, 61 (3):694–710, 2013. doi: 10.1287/opre.1120.1149. URL `http://pubsonline.informs.orghttp://dx.doi.org/10.1287/opre.1120.1149`.

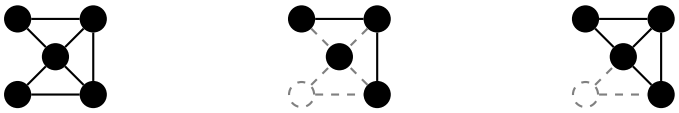
Appendices

A Graph Definitions

Definition A.1 (Graph)**.** A *graph*, $G = G(N, E)$, is made up of: a set of *nodes* (also called *vertices* or *points*), N, which are places ; and a set of *edges* (also called *arcs* or lines), E, which are connections between places, so elements of E must be two-element subsets of N.

Definition A.2 (Subgraph)**.** A graph $Q' = (N', E')$ is said to be a *subgraph* of $Q = (N, E)$ if $N' \subset N$ and $E' \subset E$.

A subgraph is said to be *induced* by N' (or *edge-preserving*) if E' contains all edges (from E) that have both end points in N'.



(a) Q (b) Q_1 (c) Q_1

Figure A.1: Q_1 is a subgraph of Q. However it is not induced as it is missing possible edges connecting nodes that existed in Q. Q_2 shows the induced subgraph on the chosen set of nodes.

Definition A.3 (Walk,Path,Trail,Cycle)**.** A sequence of nodes $(n_0, n_1, ..., n_l)$ is a *walk* of length l if $e_{n_i,n_{i+1}} \in E \ \forall i = 0, ..., l-1$. Corresponding to a walk is the sequnece of l edges $(e_{n_0,n_1}, e_{n_1,n_2}, ..., e_{n_{l-1},n_l})$.

A walk becomes a trail if each edge in the walk is distinct, i.e $e_{n_i,n_{i+1}} \neq e_{n_j,n_{j+1}} \forall i \neq j$. A trail becomes a path if each node in the walk is distinct (except possibly the start and final node), i.e $n_i \neq n_j \forall i \forall i < j \geq l-1$.

A walk,trail or path is said to be *closed* if the start and end nodes are the same. A *cycle* is a closed path with length, $l \geq 3$ (with the special case of $l = 3$ being called a *triangle*).

Definition A.4 (Hamiltonian cycle)**.** A *Hamiltonian cycle* is a cycle which contains every node on the graph, i.e it is a cycle of length $l = |N|$. A graph that exhibits a Hamiltonian cycle is called *Hamiltonian*

Example A.5. For the graph Q as in Figure 4:

- An example of a walk is $(1, 2, 1, 5, 4, 2)$
- An example of a trail is $(1, 2, 5, 3, 4, 5, 1)$

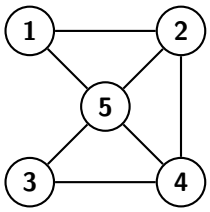


Figure 4: Graph, Q

- An example of a path is $(1, 2, 4, 3)$
- An example of a Hamiltonian cycle is $(1, 2, 4, 3, 5, 1)$

Hence we would call the graph Q Hamiltonian.

Definition A.6 (Complete graphs)**.** The *complete graph*, K_n, is a graph of n nodes, in which all edges are present, i.e $e_{i,i'} \in E \; \forall i, i' \in N$.

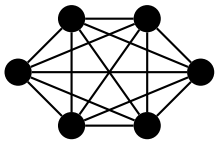
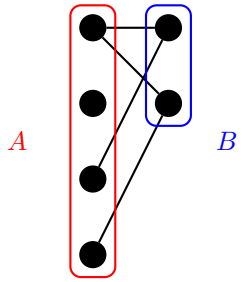


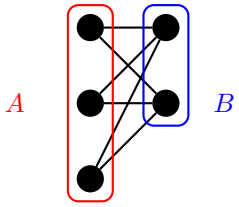
Figure A.2: The complete graph of 6 nodes,K_6.

Definition A.7 (Bipartite)**.** A graph is said to be *bipartite* if $N = A \cup B$, where $A \cap B = \emptyset$,and $e_{i,i'} \notin E \; \forall i, i' \in A$, $e_{i,i'} \notin E \; \forall i, i' \in B$.

Definition A.8 (Complete bipartite)**.** The *complete biparite graph*, $K_{a,b}$, is a bipartite graph of $a + b$ nodes (where $|A| = a$,$|B| = b$), in which all edges are present, i.e $e_{i,i'} \in E \; \forall i \in A \forall i' \in B$ and $e_{i,i'} \in E \; \forall i \in B \forall i' \in A$.



(a) Q



(b) $K_{3,2}$

Figure A.3: A.3a is an example of a bipartite graph, Q. A.3b is the complete bipartite graph with set sizes of 3 and 2.

Definition A.9 (Subdivision,Smoothing)**.** A *Subdivision* (or *expansion*) of a graph, G, is a new graph G' which is made by subdividing a chosen edge. The subdivision of an edge $\{u, v\}$ yields a graph with a new node w and the splitting of the edge $\{u, v\}$ into $\{u, w\}$ and $\{w, v\}$.

The reverse process is called *Smoothing* of a graph, G, is a new graph G' which is made by smoothing between two nodes. The smoothing out of a node pair (u, v), with $d(u, v) = 2$ and with w between them, then w is removed along with the edges $\{u, w\}$ and $\{v, w\}$, then the edge $\{u, v\}$ is placed to connect u and v.

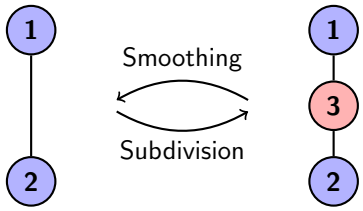


Figure A.4: Subdivision and Smoothing of the edge $\{1, 2\}$

Definition A.10 (Edge-vertex dual)**.** A *edge-vertex dual* of a directed graph G, called $EV(G)$, is made of a vertex set $V_{EV(G)} = E_G$ and whose edge set is made up of a directed edge between $e_1, e_2 \in V_{EV(G)}$ if in G the edge e_1's head meets the tail of e_2 at some node.

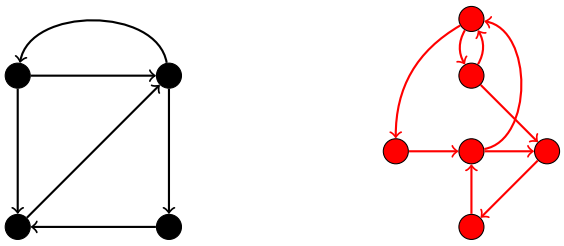


Figure A.5: A graph G and its directed edge-vertex dual, $EV(G)$

B Optimal Solution for a Random Attacker

C Optimal Solution for Random Attacker with Local-Observations