Second Year Report

Thomas Lowbridge
School of Mathematical Sciences
University of Nottingham

June 20, 2018

Contents

Summary of report

In section 1 we provide an overview of the current work done on security problems, which focus on a patroller attempting to stop some malicious entity. We will also provide an in-depth review of [3] and [11] upon which our contributions in sections **??** and 4 will be based upon.

In section **??** we continue last years initial work and provide new optimal strategies for the patroller as well as correcting an optimal attack strategy which solves a small issue with Lemma 9 in [3]. We further develop the idea that led to this correction in order to allow us to solve a wider class of graphs.

In Section 3 we analyse the work done in [3] more closely, particularly looking at the issue with having only deterministic attack times. This is in an aim to provide evidence that the errors in our work in Section 4 are exhibited in the original problem, and that it is a case of too much knowledge in the Lagrangian relaxation. We propose multiple solutions to the problem and carry out numerical experiments.

In Section 4 we build on the work done in [11] by introducing the concept of the patroler observing suspicous behaviour of arriving attackers and try to incorporate this information in an effort to develop a similar heuristic.

Finally in Section 5 we conclude the report with a list of uncompleted work as well as a brief introduction to ideas for future work and extensions to our security problems.

We leave most proofs to the Appendix, in which we also provide some graph definitions for those unfamiliar with common graph terminology.

1 Literature Reviews

1.1 Overview of Security Problems

Security problems arise in many real world scenarios, such as a security guard patrolling a museum, police patrolling a city and military personal guarding bases and borders. At the core of all these problems is the need to deploy resources to locations in order to find undesirable activity. When resources are scarce but can be used continuously one may suggest a patrol of the locations, which means the structure of the locations and how they can be traversed is of key importance. While most of the time this idea of traversing is physical, be it a guard walking around a museum or a drone flying of a region, it is not necessarily true, it may be a guard flicking through live-video feeds.

Early work in the area focused on police patrols in urban areas ([10],[7],[6]), later research focused on patrolling rural areas ([4]) and highways ([17]). These early works relied on the fact that the crime rate was known at the different locations and that they remained constant while the patrol was happening and involved randomizing the patrols in order to maximize the probability of intercepting a crime in progress.

While these are a strategic problem from only the police force's point of view, one may wish to introduce a strategic problem for the attacker, making the problem a game theoretic one, while differential game formulations exist ([8]) these tend to focus on a dynamic (and often strategic) process of mutual adjustment, rather than confronting the problem of a scheduler who has has to determine the path of the patrol to take beforehand.

Game-theoretic analyses have been done in studies of counter-terrorism actions ([5],[13]), to find how a patroller should randomize their patrol, it has been shown that the obvious that given n targets spend $\frac{1}{n}$ of the available patrol time on every target is not necessarily a good idea([9]). The problem has been formulated as a Stackleberg (leader-follower decisions) type game, instead of a simultaneous decision one, in which heuristics have been found that near optimally solve the problem ([15]).

A finite time ,simultaneous decision, game-theoretic formulation, based on a search game has been formulated and is called the patrol game ([3]). A search game involves a searcher who aims to minimize the time to find a hider, who doesn't want to be found, in some region ([1])). Section 1.2 reviews the formulation and results of the patrol game, which uses discrete time and the same deterministic attack times. While general tools were found, only exact solutions to certain classes of graphs were found and in particular only a partial solution to the line graph was found, which was later completed ([14]). A continuous version time version of the patrol game has also been formulated and solved for the line graph, agreeing with the discrete case ([2]).

Following this work, a infite time horizon patrol problem was formulated for generic attack times which can differ between nodes ([11]), this problem is reviewed in Section 1.3. By using work related to indices developed for multi-arm bandits, they manage to develop near optimal heuristics for the patrol problem. They then move to the infinite time horizon patrol game, by introducing a strategic attacker, and again find near optimal heuristics for the problem. The patrol problem and game were later extended to include the idea of overlooking an attack instead of capture being ensured and again near optimal heuristic were developed ([12]).

Our work will focus on extending the work done in [3] and [14] to get an exact answer for more general graphs, with the aim of solving the exact game for trees. We develop a similar theory as in [11] to deal with a problem, where the patroller may observe some suspicious behaviour from attackers and incorporates this information into their decision.

1.2 Review of Strategic Patrolling games

This section summarizes the key work from [3] and their work on Patrol games. Providing a baseline for future work done in section 2

1.2.1 Game set-up

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 always 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

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

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 $\mathcal{P} = (P(W_i, A_j))_{i \in \{1,...,|\mathcal{W}|\}, j \in \{1,...,|\mathcal{A}|\}}$ (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 $V = V(Q, T, m) \equiv \max_{\boldsymbol{\pi} \in \Pi} \min_{\boldsymbol{\phi} \in \Phi} P(\boldsymbol{\pi}, \boldsymbol{\phi}) = \min_{\boldsymbol{\phi} \in \Phi} \max_{\boldsymbol{\pi} \in \Pi} P(\boldsymbol{\pi}, \boldsymbol{\phi})$, in which we say that $(\boldsymbol{\pi}, \boldsymbol{\phi})$ is a optimal *Nash equilibrium*, and when needed we distinguish between the one-off and period 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 [3] 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 section as it provides an upper bound for the periodic game.

1.2.2 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.

Definition 1.2 (Edge-preserving subgraph)**.** A subgraph, $Q' = (N', E')$ of Q is called edge-preserving if $n_1, n_2 \in Q \cap Q*$ then if $(n_1, n_2) \in E \implies (n_1, n_2) \in E'$

Meaning for an edge-preserving subgraph,we only need to pick the nodes and the edges are mandated by the original graph.

Lemma 1.3 (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 $Q = \bigcup\limits_{i=1}^{k} Q_i$ then*

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

Furthermore in the case of a disjoint decomposition equality is reached

The above provides a solution to build disjointly decomposable graphs, so it is only worth studying connected graphs.

Example 1.4. For Q as seen in Figure 1.1. Consider when $m = 3$, the decomposition of Q into the graphs $Q_1 \equiv L_2$ and $Q_2 \equiv L_3$. $V_1 = V(L_2) = 1$ as alternating between 1 and 2 can catch every attack. $V_2 = V(L_3) = \frac{3}{4}$ (as seen in [3]).

Then we can get the bound $V \geq \frac{1}{\frac{1}{V_1}+\frac{1}{V_2}} = \frac{1}{\frac{1}{1}+\frac{3}{4}} = \frac{4}{7}$.

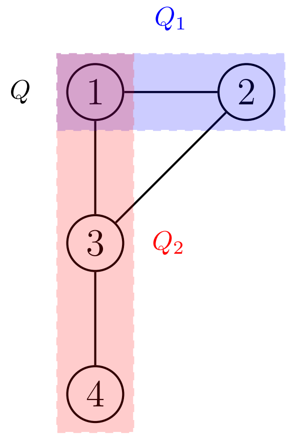


Figure 1.1: Decomposition of Q into Q_1 and Q_2.

Simplification(Patroller and Attacker):

Definition 1.5 (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*.

Definition 1.6 (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 1.7 (Simplification)**.** *If Q' is a simplified version of Q then $V(Q') \geq V(Q)$*

Simplification allows us to get bounds for both the patroller and attacker, depending on whether Q' or Q is the graph we are working on.

Example 1.8. For Q as seen in Figure 1.2, when $m = 3$, the simplification of the graph by identifying nodes 1 and 2 simplifies Q to $Q' = L_3$. Hence we can get the bound that $V(L_3) \geq V(Q)$ so $V(Q) \leq \frac{3}{4}$

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

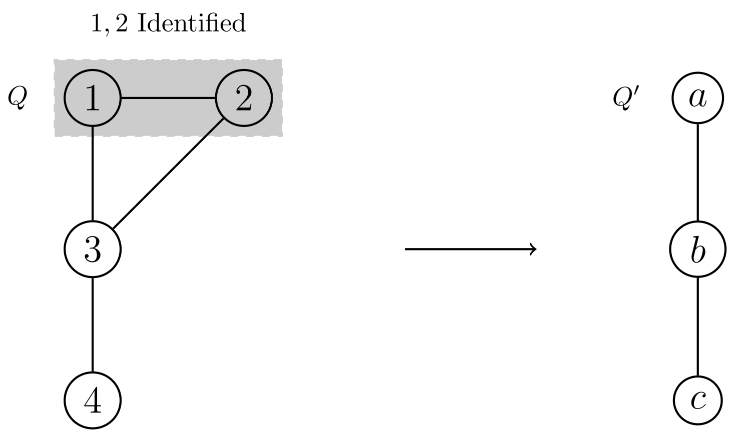


Figure 1.2: Simplifcation of Q to Q' by identification.

Lemma 1.10. *The attacker playing equally likely at a pair of diametrical for a random time interval is called the diametric attack, and gives* $V \leq \max\left\{\frac{m}{2\bar{d}}, \frac{1}{2}\right\}$

In section **??** we will present an issue with this lemma.

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* (for attack time, m) if any patrol intercepting an attack at i cannot also intercept an attack at i'.

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.

For the one-off game being independent is equivalent to $d(i, i') \geq m$ and for the Periodic game it is equivalent to $d(i, i') \geq m$ and $2d(i, i') \leq T$(due to returning to start).

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)**.**

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

1.2.3 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, that is a walk which visits all nodes only once, returning to the start. Two classes of such graphs are cyclic graphs, C_n and the complete graphs, K_n. While Hamiltonian graphs can exhibit more than one Hamiltonian cycle we shall assume that we have selected one arbitrarily. 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 bound (Lemma 1.1) , provides the solution $V = \frac{m}{n}$.

Line:

A Line Graph, L_n, is a graph consisting of n nodes with $n-1$ edges connecting the nodes in a straight line. While the line graph is complicated to solve it has been done across two papers, [3] and [14]. The solution required the division of the (n, m) space into sub-regions (inside which different strategies are adopted by the attacker and patroller). We highlight the regions for comparison to solving our bigger class of graphs later.

- $S_1 = \{(n, m) \,|\, m > 2(n-1)\}$ for $V = 1$
- $S_2 = \{(n, m) \,|\, n-1 < m \leq 2(n-1)\}$ for $V = \frac{m}{2(n-1)}$
- $S_3 = \{(n, m) \,|\, m = 2, n \geq 3\}$ for $V = \frac{1}{\lceil \frac{n}{2} \rceil}$
- $S_4 = \{(n, m) \,|\, m = n-1, \text{ or } m = n-2 \text{ and } m \geq 3 \text{ odd}\}$ for $V = \frac{1}{2}$
- $S_5 = \{(n, m) \,|\, m \leq n-3, \text{ or } m = n-2 \text{ and } m \geq 3 \text{ even}\}$ for $V = \frac{m}{n-1+m}$

However S_2 uses the diametric attack, so will exhibit the issue discussed in Section **??** and will be corrected.

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 $K_{a,b}$, in which all combinations of nodes in A and B are adjacent.

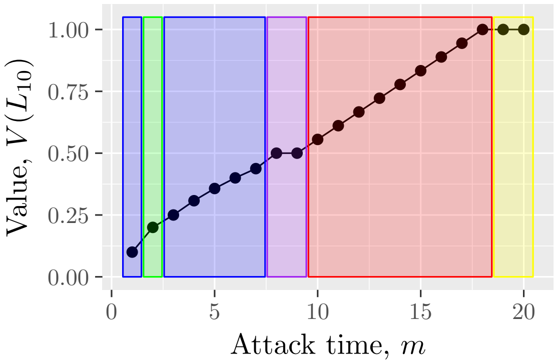


Figure 1.3: Value of L_{10} for S_1,S_2,S_4,S_5,S_3

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 if m is odd).

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 Review of patrolling problem with random attackers

This section summarizes the key work from [11] and their work on Patrolling games with random attackers. Providing a baseline for future work done in section 4

1.3.1 Problem set-up

A patrolling problem with random attackers, $G = G(Q, \boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{c})$ is a minimizing problem for the patroller. The parameters of the problem 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, $\boldsymbol{c} = (c_1, ..., c_n)$

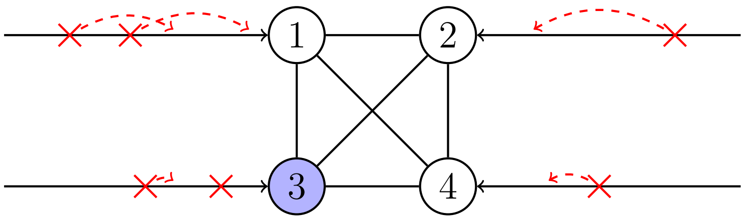


Figure 1.4: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

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 decides which node to move to and moves there taking unit time to arrive(which can be the current node).

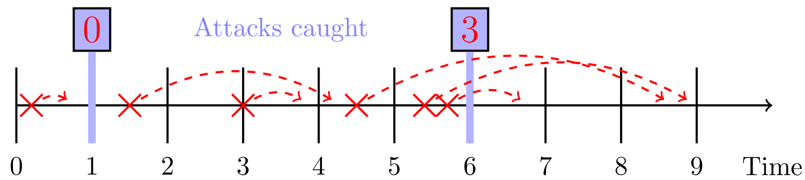


Figure 1.5: Example for a given node, when the patroller visits at times, $1, 5$

We can formulate such as problem as a Markov Decision Process(MDP) with state space, $\Omega = \{\boldsymbol{s} = (s_1, ..., s_n) \mid s_i = 1, 2, ... \text{ for } i = 1, ..., n\}$, where s_i denotes the time 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 $l(\boldsymbol{s}) = \operatorname{argmin}_i s_i$. The available decisions from state $\boldsymbol{s}$ are $\mathcal{A}(\boldsymbol{s}) = \{j \mid A_{l(\boldsymbol{s}),j} = 1\}$ 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, the process is a Markov Chain(MC) and hence the patroller's problem is justified as a MDP.

The patroller incurs costs for all attackers able to complete their attacks. When the decision to move to node i is made the cost incurred for the next time period is $C(\boldsymbol{s}, i) = \sum_{j=1}^{j=n} C_j(\boldsymbol{s}, i)$, where $C_j(\boldsymbol{s}, i)$ is the cost at node j choosing to move to node i in the next time period. We have that

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

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 Section 8.10.1 in [16]), so we bound the state space to make it finite.

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

We now see that the cost function remains constant, at $c_j \lambda_j$, 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 know, that due to the finite state space that we can just focus on the class of stationary, deterministic policies $\Pi = \{\pi : \Omega \to N \,|\, \pi(\boldsymbol{s}) \in \mathcal{A}(\boldsymbol{s})\}$ (See Theorem 9.1.8 in [16]). So we wish to solve

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

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 $C^{\text{OPT}}(\boldsymbol{s}_0)$ does not depend on the initial state and so $C^{\text{OPT}} = C^{\text{OPT}}(\boldsymbol{s}_0)$ is the same for all initial states.

If we set $c_i = 1 \ \forall i$ then we can interpret 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. How to formulate both approachs is left to Appendix D. However we should note that such methods are slow and only computable for small graphs, therefore we seek to create a near-optimal heuristic policy that can be run in a relatively short time.

1.3.2 Problem relaxation

We first relax the issue 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, which need not be adjacent. We will denote this 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 $\pi^n \equiv \pi$, so $C^{\text{OPT}} \geq C^{\text{MN}} \equiv \min_{\pi^n \in \Pi^{\text{MN}}} \sum_{i=1}^{n} V_i(\pi^n)$.

Like before π^n will induce a patrol pattern, $\psi_{\pi^n}^k$ 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

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

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 to the corresponding set of policies,

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

Again, $\exists\, \pi^n \in \Pi^{\mathrm{TR}}$ s.t $\pi \equiv \pi^n$, so $C^{\mathrm{OPT}} \geq C^{\mathrm{TR}} \equiv \min_{\pi^n \in \Pi^{\mathrm{TR}}} \sum_{i=1}^{n} V_i(\pi^n)$.

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 $\mu_i(\pi^n)$ instead.

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

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

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

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.* The Lagrangian relaxation separates the problem n individual node problems, where node i's problem is

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

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 or wait. 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 D

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 [11]. 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 Patrolling games

2.1 Problem and correction to line graph strategy

We will now look at the problem with the diametric attack, $V \leq \frac{m}{2\bar{d}}$ (when $\bar{d} < m \leq 2\bar{d}$), consider a patroller who oscillates between the two diametric points. We now want to see how many attacks the patroller can capture out of the possible $2(T - m + 1)$ the attacker is making.

As a counter-example consider

Example 2.1 (Counter-example for diametric bound). As a counter-example consider the game $(Q = L_5, T = 20, m = 6)$ so $m = 6 > \bar{d} = 4$, then under the diametric attack the attacker has 30 attacks, starting at $0, ..., 14$ at nodes 1 and 5. If the patroller oscillates between diametrics points they capture 1+5+6+6+4 = 22 attacks out of $2(20-6+1) = 30$ attacks, for a capture chance of $V = \frac{11}{15}$, which is greater than diametric bound $\frac{3}{4}$ which provides an upper bound, and hence is wrong.

Because of the counting of attacks, it is possible to delay the patroller at the start to still get a better number of captured attacks, e.g. delaying in the counter example to leave at time 2 instead of 0 gives $3+6+6+6+2 = 23$. In fact it is best to delay yourself to leave at time $m - \bar{d}$. This can be seen in Appendix B.1

This means we will capture

$$\underbrace{m-\bar{d}}_{\text{Waiting initially}} + \underbrace{\left(m \times \left(\left\lfloor \frac{T-2m+1}{\bar{d}} \right\rfloor + 1\right)\right)_+}_{\text{Visits which get exactly } m \text{attacks}} + \underbrace{\left(T - \left(m - 1 + \left(\left\lfloor \frac{T-2m+1}{\bar{d}} \right\rfloor + 2\right)\bar{d}\right)\right)_+}_{\text{Penultimate and Final node visits}}$$

The key point is that the upper bound given by the diametric attack is dependent on T. An example is given in Figure 2.1 which shows that for some T as in Lemma 2.2 and as the finite time horizon becomes infinite the corrected bound reaches the suggested bound in [3].

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.*

For proof see Appendix B.2

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 (Time-limited diametric attack bound)**.** *When $T \geq m + \bar{d} - 1$, the diametric bound $V \leq \max\{\frac{1}{2}, \frac{m}{2\bar{d}}\}$ is valid.*

For proof see Appendix B.3

2.2 New Tools

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

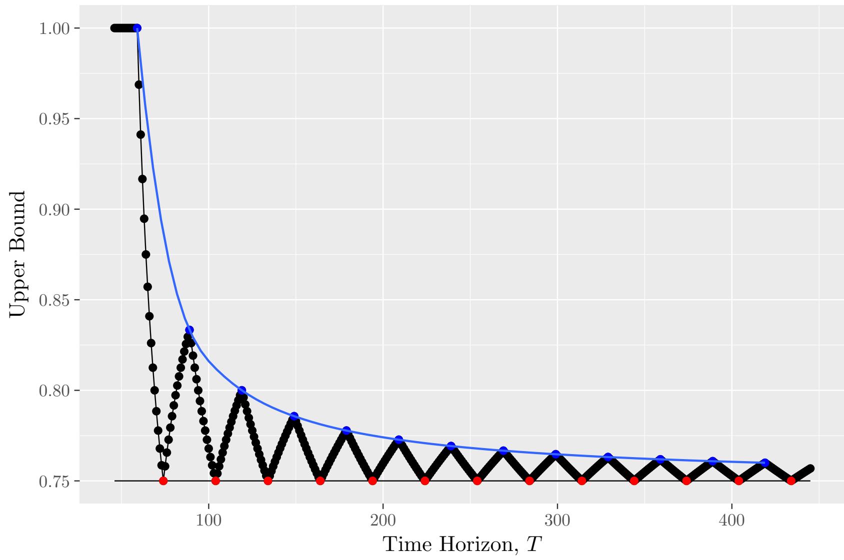
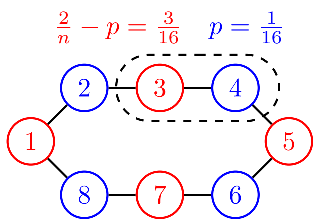


Figure 2.1: Best Upper Bound achievable under the diametric strategy

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 B.4).

We now look at extending the idea of the diametric attack. First we notice that we are not forced to use the graphs diameter, $\bar{d}$, we can use any distance between two selected nodes, d. We replacing $\bar{d}$ with d we will get a two node distance attack, giving us a bound of $V \leq \max\{\frac{1}{2}, \frac{m}{2d}\}$. However this is only possibly worse than using the graphs diameter, so this is not useful.

A useful extension, would be not using two nodes but using multiple points each the same distance apart from each other mutually.

Definition 2.7 (Polygonal attack)**.** A *d-polygonal attack* is an attack at a set of nodes $D = \{i \in N \,|\, d(i,i') = d\}$ at the time intervals $I, I+1, ..., I+d-1$ (for a chosen initial I)

We can also consider having the points *at least* the same distance apart

Definition 2.8 (Uneven polygonal attack)**.** A *d-Uneven polygonal attack* is an attack at a set of nodes $D = \{i \in N \,|\, d(i,i') \geq d\}$ at the time intervals $I, I + 1, ..., I + d - 1$ (for a chosen initial I)

Note. Just like the time-limited diametric attack, the (uneven) polygonal attack is only feasible if $T \geq m + d - 1$.

The idea behind this attack is very similar to the timed diametric attack.

Lemma 2.9. *When $T \geq m+d-1$ and a set D as in the d- (uneven) polygonal attack, the bound $V \leq \max\{\frac{1}{|D|}, \frac{m}{|D|d}\}$ is valid*

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

The line and star graphs provided a good starting point for attempting to solving the problem with a general tree graph. If a more general version of the star graph can be solved, it may provide better bounds on a general tree.

The idea is to extend the star graph to a more general graph which is a mix between the line and the star, by extending the length of the branches (at first just one branch). This may better model a tightly packed region to search and another that is far away, consider the example of a small town and larger city connected by a road.

Definition 2.10 (Elongated Star Graph)**.** The *Elongated Star Graph*, S_n^k is made from S_n, by performing subdivision on one of the edges repeatedly k times, so that one of the external nodes is now $k+1$ away from the centre.

The labelling will be done as in Example Figure 2.2 and we will from now assume that $n \geq 3$, as otherwise we are just dealing with the line.

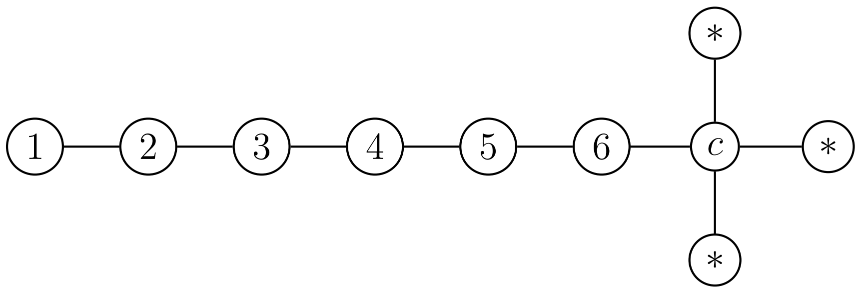


Figure 2.2: Labeling on the graph S_4^5.

To start our analysis of this graph, we can look at an expanded graph which can simplify down to our extended star graph. Consider the cyclic graph $C_{(2k+1)+(2n-1)} = C_{2(n+k)}$, we can simplify this graph by node identifying. The identifying map is one such that we identify nodes i to $2k+2-i$ for $i = 1, .., k$ and identify nodes $2k+2, 2k+4, .., 2k+2n$.

Definition 2.11 (Random Oscillation)**.** The *Oscillation* on S_n^k is any embedded Hamiltonian Patrol on $C_{2(n+k)}$ under the simplification above. The *Random*

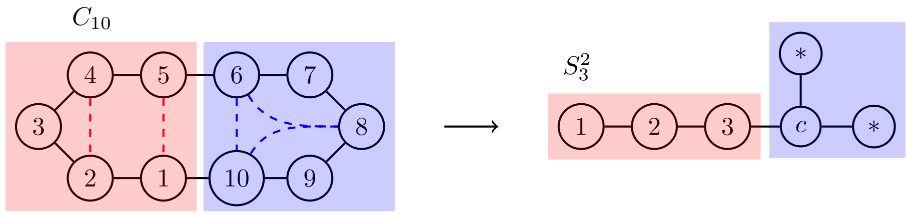


Figure 2.3: C_{10} can be simplified to S_3^2 by node identifying.

Oscillation on S_n^k is the embedded Random Hamiltonian Patrol on $C_{2(n+k)}$ under the simplification above.

Lemma 2.12. *For $m < 2(n+k)$ following the Random Oscillation,*

$$V(S_n^k) \geq V(C_{2(n+k)}) = \frac{m}{2(n+k)}$$

and if $m \geq 2(n+k)$ then $V(S_n^k) = 1$, achieved by any Oscillation.

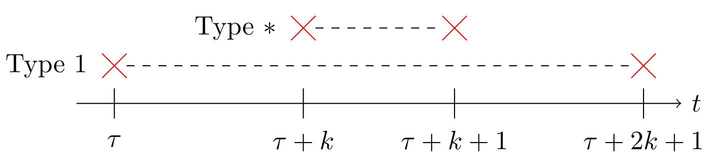
Hence we have the solution in $m \geq 2(n+k)$, so we can now restrict ourselves to $m < 2(n+k)$.

We could now consider applying the time-limited diametric attack to get bounds on the problem, $\bar{d} = n-1$ gives the bound $V \leq \max\{\frac{1}{2}, \frac{m}{2(n-1)}\}$, however this bound is not tight and this is because we are not utilising all $*$ type nodes.

We suggest using all $*$ type nodes and node 1, and assign some probability of attacking a node equivalent to the distance from the centre, c. However it turns out that the timing must also change if we wish to attack nodes with uneven probabilities. We create the time-delayed attack.

Definition 2.13 (Time-delayed attack)**.** Let the *time-delayed attack*, be the attack that attacks at the extended node labeled 1 with probability $\frac{k+1}{n+k}$ and a particular normal node labeled $*$ with probability $\frac{1}{n+k}$.

If node 1 is chosen have the attack choose probability intervals with equal probability at the times $I, I+1, ..., I+2k+1$ for some I (i.e starting attacks at $\tau, \tau+1, ..., \tau+2k+1$). If a $*$ node is chosen start the attacks at the times $I+k, I+k+1$ with equal probability.



Lemma 2.14. *When $T \geq m+2k$, the analogous 'diametric' bound $V \leq \max\{\frac{k+1}{n+k}, \frac{m}{2(n+k)}\}$ holds*

With this analogous 'diametric' we have a solution as long as we are in the range where $m \geq 2(k+1)$ so the lemma gives $V \leq \frac{m}{2(n+k)}$.

Lemma 2.15 (Solution in $m \geq 2(k+1)$)**.** *By the attacker using the stage-delayed attack and the patroller using a random oscillation patrol, we achieve the value, when* $2(k+1) \leq m \leq 2(n+k)$

$$V = \frac{m}{2(n+k)}$$

Hence we have a solution for 2 regions $m > 2(n+k)$ and $2(k+1) \leq m \leq 2(n+k)$.

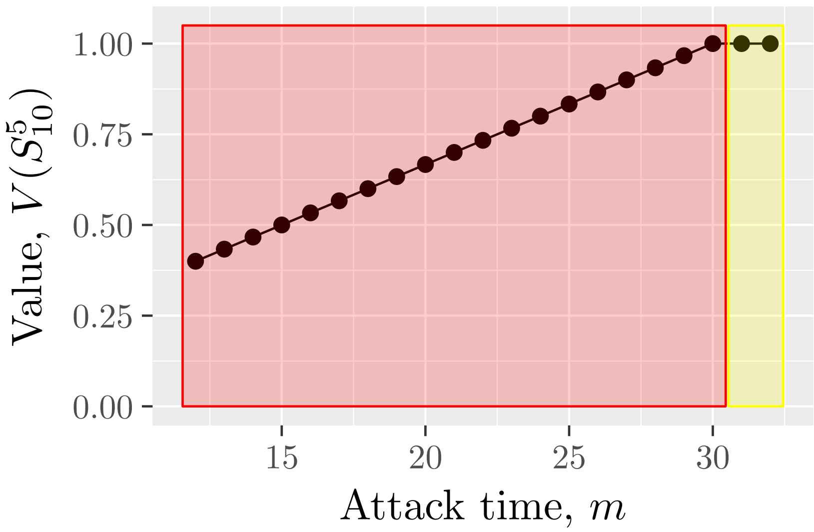


Figure 2.4: Value of the Star Graph, S_{10}^5

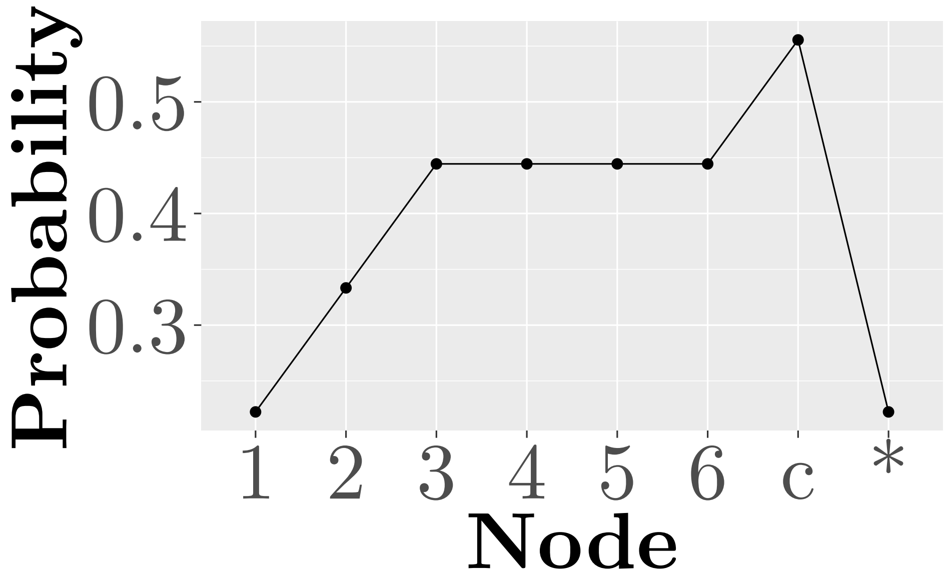
We then seek solutions in the region $m < 2(k+1)$. However in this region we are below the Random Oscillation bound for the patroller and we can suggest some improvement as in [14] from the embedded Hamiltonian bound. To see the issue and why the Random Oscillation can be improved we will look at the probability of interception under the strategy.

If the patroller is performing a Random Oscillation, then for a pure attack at node i the probability of capture is given by (derivation in appednix C.1),

$$w(i) = \begin{cases} \frac{\min(m+2(i-1),2m)}{2(n+k)}, \text{ for } i \leq \frac{n+k}{2}+1, \\ \frac{\min(m+2(n+k+1-i),2m)}{2(n+k)}, \text{ for } i > \frac{n+k}{2}+1, \\ \frac{\min(m+2(n-1),nm)}{2(n+k)}, \text{ for } i = c, \\ \frac{m}{2(n+k)}, \text{ for } i = *. \end{cases} \tag{2}$$

We will call $w(i)$ the probability of *interception* at node i.

It is very clear that the issues are towards the end of the graph, where the returns, under the Random Oscillation do not provide adequate coverage, hence



Example Figure 2.2: Interception probabilities of S_4^5 when $m = 4$.

we may wish to improve these end points of the graph. To do so we will introduce the idea of cycles which are played with guarantee capture of all attacks at noes within their cycle.

Definition 2.16 (End-ensuring cycle)**.** Define a cycle of length m (if even) or $m-1$ (if odd) to be *End-ensuring* if one of the points along the cycle is a leaf node. Define the *Half-length* of such cycles to be $\hat{m} = \left\lfloor \frac{m}{2} \right\rfloor$.

In [14] the authors improved nodes with poor interception probabilities by introducing two end-ensuring cycles. We shall do the same, though now more consideration needs to be taken on how to place these End-ensuring cycles.

We first classify nodes into types, we partition the node set, N, into

- Left nodes, $L = \left\{ i \,|\, i \le \left\lfloor \frac{m}{2} \right\rfloor + 1, i \le k+1 \right\}$
- Middle nodes, $M = \left\{ i \,|\, \left\lfloor \frac{m}{2} \right\rfloor + 2 \le i \le n + k - \left\lfloor \frac{m}{2} \right\rfloor, i \le k+1 \right\}$
- Right nodes, $R = \left\{ i \,|\, i \ge n + k + 1 - \left\lfloor \frac{m}{2} \right\rfloor, i \le k+1 \right\}$
- Star node, $S = \{c, *\}$

Note. The set R is empty if $\left\lfloor \frac{m}{2} \right\rfloor \le n-1$. The set M is empty if $\left\lfloor \frac{m}{2} \right\rfloor \ge k-1$ or $\left\lfloor \frac{m}{2} \right\rfloor \ge n+k-2$.

Then $V \ge w_{\min} \equiv w_{\min}^N = \min \left\{ w_{\min}^L, w_{\min}^M, w_{\min}^R, w_{\min}^S \right\}$, where $w_{\min}^X = \min\limits_{i \in X} w(i)$. We aim to use end-ensuring cycles to improve $w_{\min}$, which means improving the worst node sets, and hence improving the random oscillation bound.

Now from equation 2 we not some properties of each node set.

- $w^L_{min} = w(1)$, as for $i \in L$ we have that $w(i)$ is increasing in i.
- $w^M_{min} = w(\lfloor \frac{m}{2} \rfloor + 2) = \frac{2m}{2(n+k)}$, as for $i \in M$ we have that $w(i) = \frac{2m}{2(n+k)}$ $\forall i \in M$.
- $w^R_{min} = w(k+1)$, as for $i \in R$ we have that $w(i)$ is decreasing in i.
- $w^S_{min} = w(*)$, as $w(c) > w(*)$.

So as long as the improvement made on L and M is non-decreasing then only the nodes 1 and $\lfloor \frac{m}{2} \rfloor + 2$ need be considered. Similarly if in R, the improvement is non-increasing then we only need to consider the node $k+1$.Finally if in S, the improvement improves node c as good as $*$ then we only need to consider $*$. Therefore we shall only consider such improvement strategies.

Similarly let $C^X_{\min}(\boldsymbol{\pi}) = \min_{i \in X} P(\boldsymbol{\pi}, i)$, where $P(\boldsymbol{\pi}, i) = \max_{I \subset \mathcal{T}} P(\boldsymbol{\pi}, [i, I])$, is the probability of capture under $\boldsymbol{\pi}$ and $C_{\min}(\boldsymbol{\pi}) \equiv C^N_{\min} = \min \left\{ C^L_{\min}, C^M_{\min}, C^R_{\min}, C^S_{\min} \right\}$. Then we seek to select $\boldsymbol{\pi}$ to get $C_{\min}(\boldsymbol{\pi}) > W_{\min} \equiv C_{\min}(\boldsymbol{\pi}_0)$, where $\boldsymbol{\pi}_0$ is the Random Oscillation strategy.

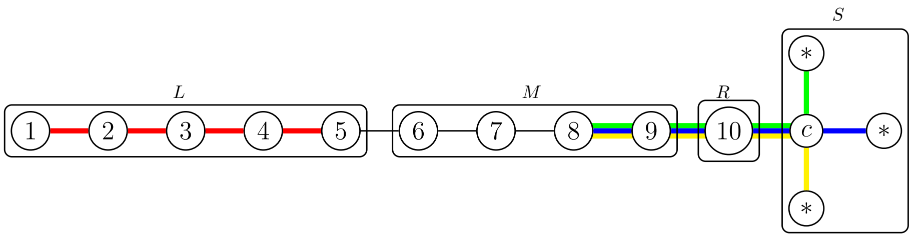
We will use P to be the probability the Random Oscillation is played and $Q = 1 - P$. We split, Q, the improvement chance into node classes, $Q = Q_L + Q_M + Q_R + Q_S$, where Q_X is the amount of probability used to improve the probability of intersection for the set X. We use q_X to be probability that the minimum of the set is improved by, so $C^X_{\min} = PW^X_{\min} + q_X$.

Multiple choices for improvement could be picked, we present first a Naive improvement, which improves it by imagining the S nodes are the ends of $|S|-1$ lines and improve as in [14]. Later we will improve this, by using combinations of these line ends.

A Naive Improvement Let $\boldsymbol{\pi} = \alpha(Q_L, Q_S)$ denote the Naive Improvement Policy. We will play the end-ensuring cycle, $\{1, ..., \lfloor \frac{m}{2} \rfloor + 1, ..., 1\}$, with probability Q_L (a non-decreasing improvement), giving $q_L = Q_L$. We will play, with probability Q_S, an end-ensuring cycle for each $*$ node, $\{*, c, ..., k+3 - \lfloor \frac{m}{2} \rfloor, ..., c, *\}$,(a non-decreasing improvement), giving $q_S = \frac{Q_S}{n-1}$.

Now we may also have improved some of the nodes in R, possibly up to node $k+3-\lfloor \frac{m}{2} \rfloor \leq n+k+1-\lfloor \frac{m}{2} \rfloor$ (as $n \geq 3$), meaning that all nodes in R are in the end-ensuring cycle $\{*, c, ..., k+3-\lfloor \frac{m}{2} \rfloor\}$, so $q_R = q_S$. If $M \neq \emptyset$ nodes in M may be improved, but the node $\lfloor \frac{m}{2} \rfloor + 2$ will not be improved as $\lfloor \frac{m}{2} \rfloor + 2 > \lfloor \frac{m}{2} \rfloor + 1$ and $\lfloor \frac{m}{2} \rfloor + 2 < k + 3 - \lfloor \frac{m}{2} \rfloor$ (as $\lfloor \frac{m}{2} \rfloor < k - 1$ when $M \neq \emptyset$), so $q_M = 0$.

Using the Naive Improvement Policy we can achieve an improvement over the Random Oscillation if $2(n+k) - nm \geq 0$ and get a bound of $V \geq \frac{2m}{2(n+k)+nm}$



Example Figure 2.3: The Naive Improvement on S_4^9 for $m = 9$. The red lines indicating the end-ensuring cycle $\{1, 2, 3, 4, 5, 4, 3, 2, 1\}$ and the other coloured lines indicating the end-ensuring cycles, for each $*$, $\{*, c, 10, 9, 8, 9, 10, c, *\}$ (as $\lfloor \frac{m}{2} \rfloor = 4$).

(or $V \geq \frac{1}{n}$ if $M = \emptyset, R = \emptyset$), by choosing optimal Q_L and Q_S (See appendix ??).

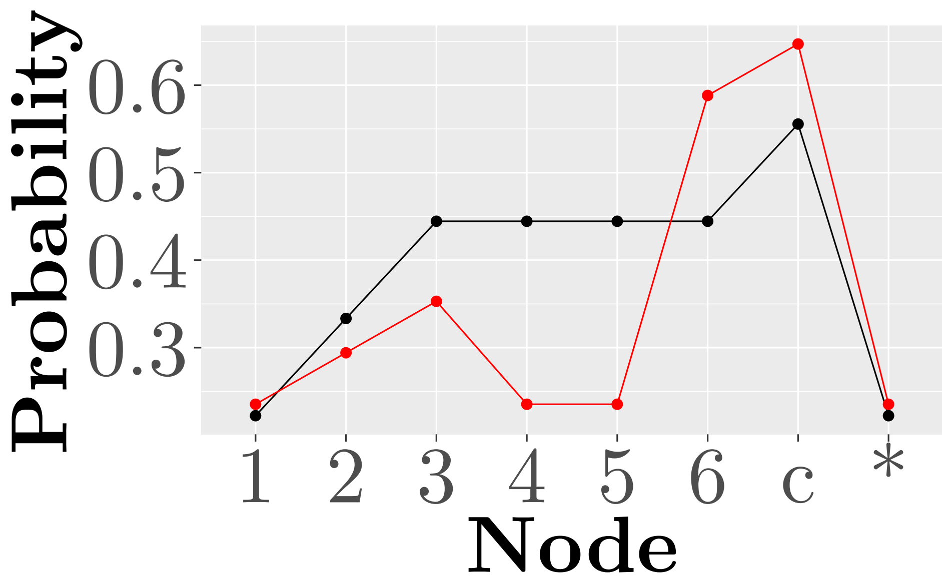


Figure 2.5: Interception probabilities of S_4^5 when $m = 4$, with the red Probabilities showing the Naive Improvement Policy $\alpha\left(\frac{2}{17}, \frac{6}{17}\right)$.

Combinatorial Improvement

Definition 2.17 (Combinatorial Improvement)**.** Let $\boldsymbol{\pi} = \beta(Q_L, Q_S)$ denote the Combinatorial Improvement Policy, which improves the sets, L and S. An end-ensuring cycle, $\{1, ..., \lfloor \frac{m}{2} \rfloor + 1, ..., 1\}$ is played with probability Q_L, so $q_L = Q_L$. Also;

Case i) If $R \neq \emptyset$ then $\lfloor \frac{m}{2} \rfloor = n + r$, for some excess r. Then form an end-ensuring cycle on the nodes $\{n+k+1-\lfloor \frac{m}{2} \rfloor, ..., k+1, c, *, c, *, ..., c, *, c, k+1, ..., n+$

$k+1-\lfloor \frac{m}{2} \rfloor\}$, this is of length $2(n-\lfloor \frac{m}{2} \rfloor+1)+2(n-1)=4n-2\frac{m}{2}=4n-2(n+r)=2n+2r=2\lfloor \frac{m}{2} \rfloor \leq m$ (so are end-ensuring). This cycle will be played with probability Q_S and improves all the nodes in R (non-increasing) and S(c is better than $*$) so $q_R=Q_R$ and $q_S=Q_S$.

Case ii) If $R=\emptyset$ then an end-ensuring cycle is formed by choosing $\lfloor \frac{m}{2} \rfloor$ $*$ nodes each equally likely. This construction is performed with probability Q_S and nodes in S (c is better than $*$). The actually improvement made is

$$q_S = Q_S \times \mathbb{P}(\text{A particular } * \text{ node is picked}) = Q_S \times \frac{\lfloor \frac{m}{2} \rfloor}{n-1}$$

To distinguish between the cases we will subscript, β, case i) β_1 and case ii) β_2.

We leave the exact analysis to Appendix **??** but note the cases bounds

- If $M=\emptyset$,$R=\emptyset$ then $V \geq \frac{\hat{m}}{\hat{m}+n-1}$
- If $M\neq\emptyset$,$R=\emptyset$ then $V \geq \frac{2m}{2(n+k)+m(1+\frac{n-1}{\hat{m}})}$
- If $M\neq\emptyset$,$R\neq\emptyset$ then $V \geq \frac{2m}{2(n+k+m)}$

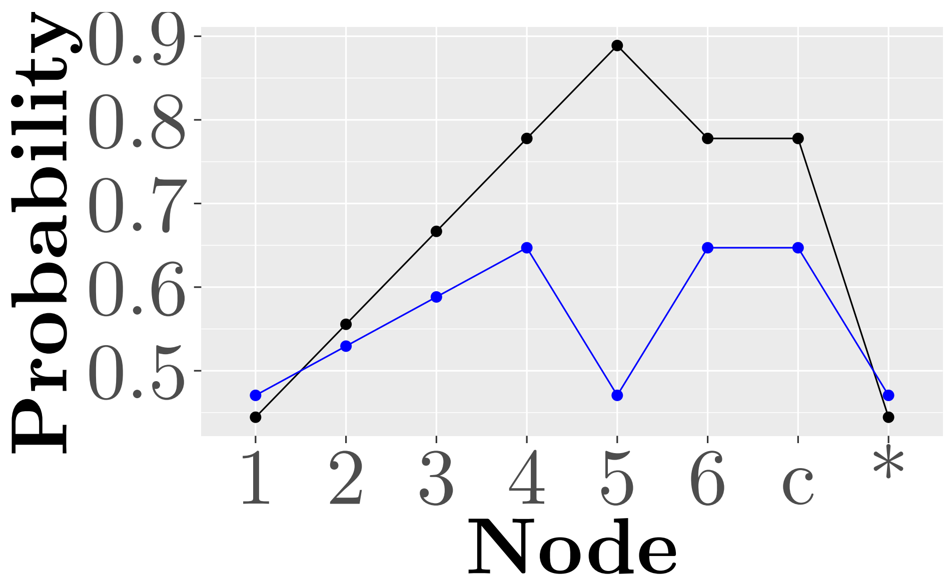


Figure 2.6: Interception probabilities of S_4^5 when $m=8$, with the blue Probabilities showing the Choosing Improvement Policy β_1 $\left(\frac{2}{13}, \frac{2}{13}\right)$.

It turns out this bound can further improved in a particular scenario, but do so we stop using end-ensuring cycles.

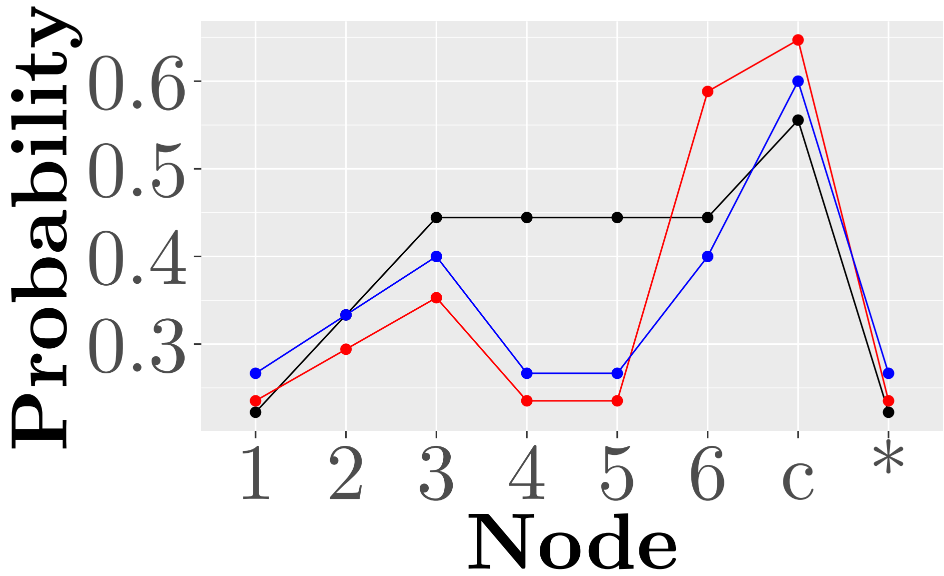


Figure 2.7: Interception probabilities of S_4^5 when $m = 4$, with the red Probabilities showing the Naive Improvement Policy $\alpha\left(\frac{2}{17}, \frac{6}{17}\right)$ and the blue Probabilities showing the Choosing Improvement Policy $\beta_2\left(\frac{1}{7}, \frac{3}{14}\right)$.

Combinatorial improvement extension We alter the idea of using end-ensuring cycles on the set of S (please note we will deal with only the case of $R \neq \emptyset$ (as that is what we need this for odd m). We will now improve the $n-1$ nodes by using a cycle of length $m+1$ (note we will be covering the case where m is odd), then when played this cycle has a probability of catching the attacker of $\frac{m}{m+1} \equiv \frac{m}{m'}$.

Now just like before we will need to play a collection of all combinations of such cycles, each cycle has a choice of $\frac{m'}{2}$ end points to visit. So we have a total of $\begin{pmatrix} n-1 \\ \frac{m'}{2} \end{pmatrix}$ and $\begin{pmatrix} n-2 \\ \frac{m'}{2}-1 \end{pmatrix}$ contain any given end point. Hence we have $\frac{\begin{pmatrix} n-1 \\ \frac{m'}{2} \end{pmatrix}}{\begin{pmatrix} n-2 \\ \frac{m'}{2}-1 \end{pmatrix}} = \frac{m'}{2(n-1)}$ Hence when we play this strategy with probability q, we improve each end with a probability of $q \times \frac{m'}{2(n-1)}$.

We will give the details and the bounds in the appendix **??** but the bound found is $V \geq \frac{2m}{2(n+k)+m+2(n-1)}$. Meaning the bound for $M \neq \emptyset, R = \emptyset$ is $V \geq \frac{2m}{2(n+k)+m+2(n-1)}$.

We will now deal with the case of $m \leq 2(n-1)$, so that all $*$'s nodes cannot be covered in one end-ensuring cycle. First we look at the $m = 2k+1, 2k$ and we note that by simplification of S_n^k into L_{k+1} and S_{n-1} we get the bound

$V \geq \frac{1}{1+\frac{2(n-1)}{m}} = \frac{m}{m+2(n-1)}$.

When $m = 2k + 1$ we can propose an attack that 'augments' the time-delayed which simply removes one attack placed at 1. That is to place attacks at node 1 starting at times, $\tau, ..., \tau + 2k$, with equal probability and $*$ nodes starting at times $\tau + k, \tau + k + 1$ (equally we could use $\tau + k - 1, \tau + k$). Similarlly if $m = 2k$ we augment it to node 1 at times $\tau, ..., \tau + 2k - 1$ and $*$ nodes at times $\tau + k - 1, \tau + k$ all equiprobable.

Conjecture 2.18. *Using the 'augmented' time-delayed we propose the attacker can get a bound of* $V \leq \frac{m}{m+2(n-1)}$ *and hence* $V(S_n^k) = \frac{m}{m+2(n-1)}$ *for* $m = 2k + 1, 2k$

The idea of the proof for the 'augmented' time-delayed attack is the same as the time-delayed attack , but with a reduced number of attacks at node 1, to make sure the patroller can only get m by waiting here.

We leave the idea of the rest of the solution, which follows a similar idea, to be introduced in Section [Need a reference]

For now we move to a more generalised star graph, where all external nodes can be any distance from the centre node.

Definition 2.19 (General star graph)**.** The *General star graph,* $S_n^{\boldsymbol{k}}$ ($\boldsymbol{k} \in \mathbb{N}^h$) is made from S_n, by performing subdivision on h of the initial edges each repeated k_i for $i = 1, ..., h$.

Note. For ease of notation we will use the above format, but will understand that using $k_i = 0$ is a valid construction, so we can image that $\boldsymbol{k} \in \mathbb{N}^n$. For ease of notation we will define $|k| \equiv \sum_{i=0}^{i=h} k_i$ and $k_{\max} \equiv \max_{i=1,2,..,h} k_i$.

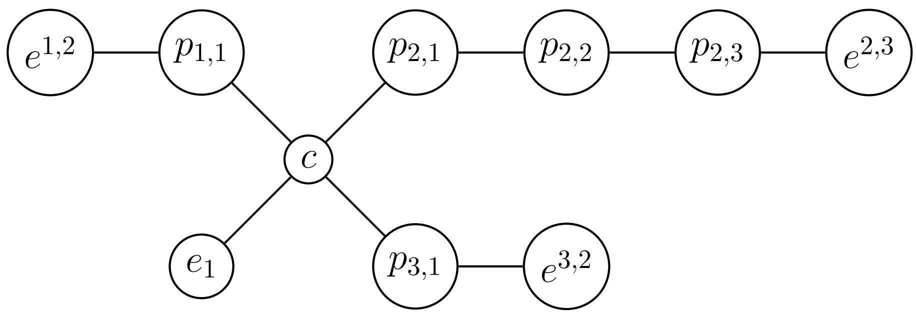


Figure 2.8: Labelling of $S_4^{1,3,1}$

To start our analysis of this graph, we will look at an expanded graph which can be simplified down to our general star graph. Consider the cyclic graph $C_{2(n+|k|)}$, we can simplify this graph by node identification to $S_n^{\boldsymbol{k}}$. The identification mapping is harder to visual from a cycle graph.

For ease of understanding look at an example as in Figure 2.9

The mapping is done as such:

- The centre is identified form nodes $1, 1+2(k_1+1), 1+2(\sum_{i=1}^{2}(k_i+1)), ..., 1+2(\sum_{i=1}^{h}(k_i+1), 1+2(|k|+h)+2), ..., 1+2(|k|+h)+2(n-h)$
- The first branch is identified from the nodes between 2 and $2(k_1+1)$ (Inclusive). $n_{1,i}$ for $i = 1, ..., k_1$ are identified by the two nodes $i+1$ and $2k_1+3-i$, the node n_{1,k_1+1} is identified by the one node k_1+2.
- The j^{th} branch is identified from nodes between $2(\sum_{i=1}^{j-1}(k_i+1)))$ and $2(\sum_{i=1}^{j}(k_i+1)))$ (Inclusive). $n_{j,i}$ for $i = 1, ..., k_j$ are identified by the two nodes $2(\sum_{i=1}^{j-1}(k_i+1))) + (i-1)$ and $2(\sum_{i=1}^{j}(k_i+1))) - (i-1)$, the node n_{j,k_j+1} is identified by the one node $2(\sum_{i=1}^{j-1}(k_i+1))) + k_j$.

This mapping gives rise to the general Oscillation

Definition 2.20 (General Random Oscillation)**.** The *oscillation* on $S_n^{\boldsymbol{k}}$ is any embedded Hamiltonian patrol on $C_{2(n+|k|)}$ under the simplification above. The *random oscillation* on $S_n^{\boldsymbol{k}}$ is the embedded random Hamiltonian patrol on $C_{2(n+|k|)}$ under the simplification above.

Lemma 2.21. *For $m < 2(n+|k|)$ following the random oscillation*

$$V(S_n^{\boldsymbol{k}}) \geq V(C_{2(n+|k|)}) = \frac{m}{2(n+|k|)}$$

and if $m \geq 2(n+|k|)$ then $V(S_n^{\boldsymbol{k}}) = 1$, achieved by any oscillation.

We will match the oscillation bound of $\frac{m}{2(n+|k|)}$, by further extending time-limited attack and time-delayed attack into the type-delayed attack.

Definition 2.22 (Node types)**.** A *type* i node, is an external node which has been extended i times. Let $k_{max} \equiv \max_{i=1,...,h} k_i$ be the maximum node type on the general extended star graph, $S_n^{\boldsymbol{k}}$.

Definition 2.23 (Type-delayed attack)**.** Let the *Type-delayed attack*, be the attack that attacks at a type i node with probability $\frac{i+1}{n+\sum_{j=1}^{h} k_j}$ $\forall i$. Choosing an attack interval I, the attack at a type i node chooses a interval from the following with equal probability: $I+(k_{max}-i), I+(k_{max}-i)+1, ..., I+k_{max}+i+1$ $\forall i$ (i.e statring attacks at a type i node at times $\tau+(k_{max}-i)+1, ..., \tau+(k_{max}+i)+1$).

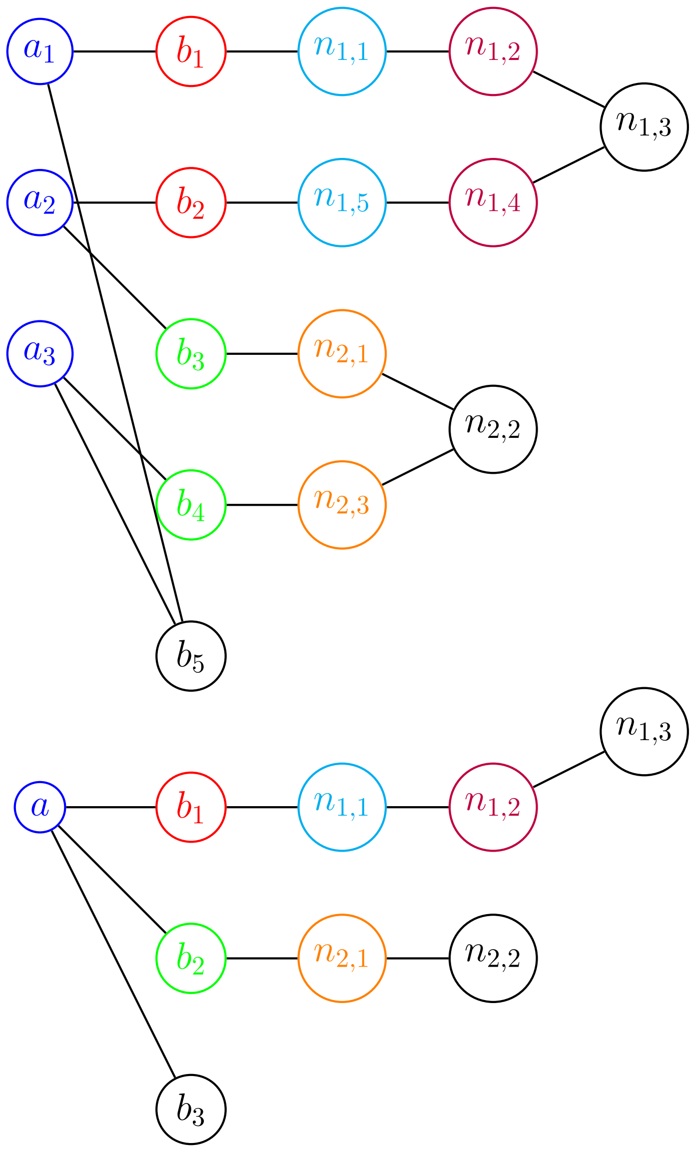
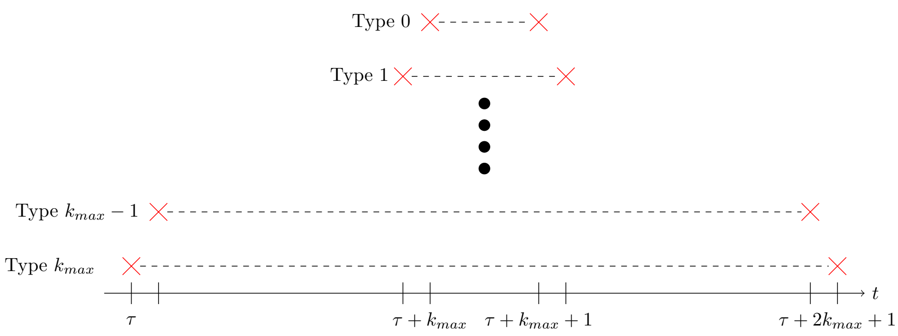


Figure 2.9: Simplification of C_{16} to $S_3^{3,2}$



Theorem 2.24. *When $T \geq m + 2k_{max}$, the analogous 'diametric' bound is given by*

$$V(S_n^{\boldsymbol{k}}) \leq \max\left\{\frac{k_{max}+1}{n+\sum_{j=1}^{h} k_j}, \frac{m}{2\left(n+\sum_{j=1}^{h} k_j\right)}\right\}$$

Proof: **??**

Corollary 2.25 (Solution in $m \geq 2(k_{max}+1)$)**.** *By the attack using the type-delayed attack and the patroller using the random oscillation we achieve the value, when $2(k_{max}+1) \leq m \leq 2(n+|\boldsymbol{k}|)$,*

$$V = \frac{m}{2(n+|\boldsymbol{k}|)}$$

and when $m > 2(n+|\boldsymbol{k}|)$ then $V = 1$.

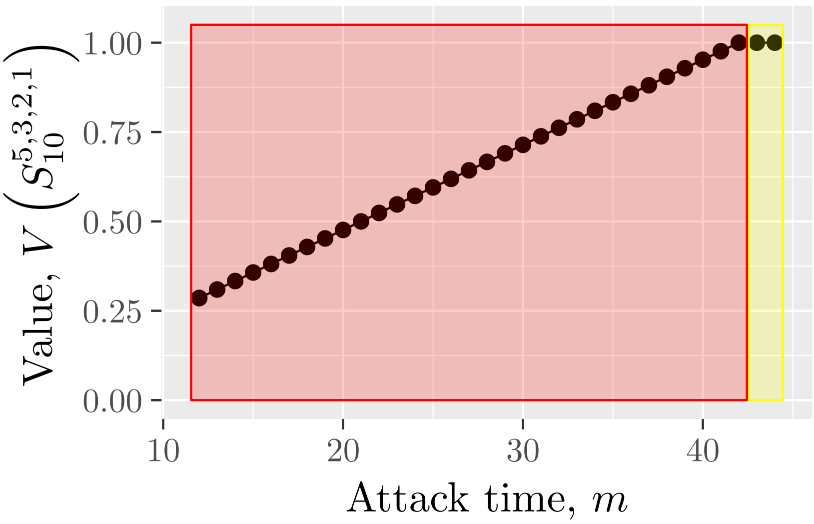


Figure 2.10: The value of the star graph, $S_{10}^{5,3,2,1}$

2.5 Joining star graphs by centralised connections

We now look at connecting by their centres the generalised star (with branches)

Definition 2.26. We define the *multi general p-star graph*, $(s_{n_1}^{\boldsymbol{k}_p}, ..., S_n^{\boldsymbol{k}_p}) \equiv \bigodot_{i=1}^{p} S_{n_i}^{\boldsymbol{k}_i}$, to be the p star graphs, $S_{n_i}^{\boldsymbol{k}_i}$ initially with disconnected centres which are now made adjacent by the introduction of connections between each combination of centres (i.e the complete graph of centres)

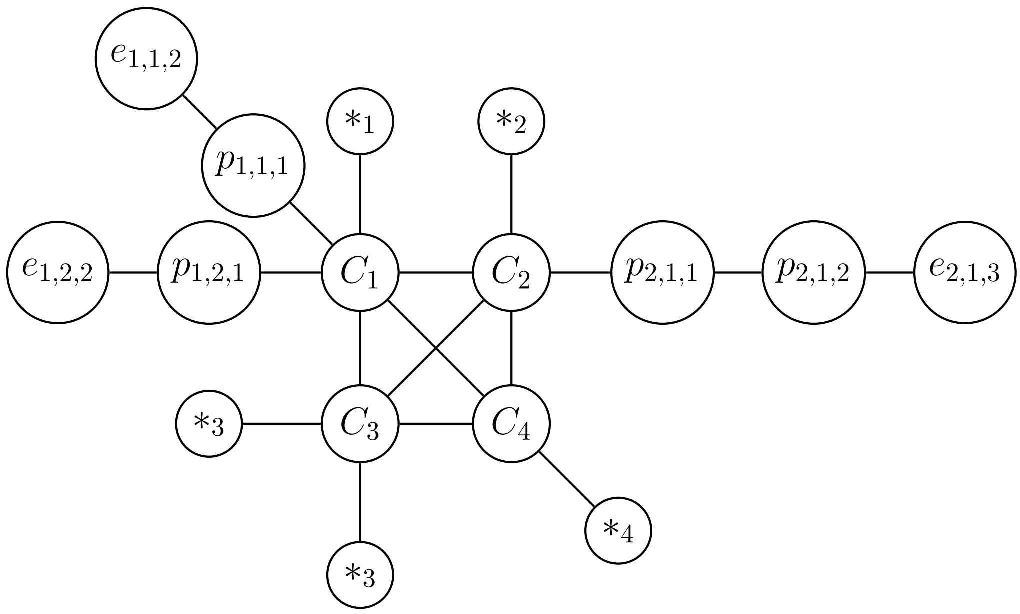


Figure 2.11: Example of labelling on $(S_3^{1,1}, S_2^2, S_2, S_1)$

Because we have just joined graphs we know some solutions to, we can consider decomposition for the patrollers decision and simplifiation for the attackers decision.

Theorem 2.27 (Separable solution)**.** *For* $2(K_{max}+1) \leq m \leq 2 \min\limits_{i=1,\dots,p}\{n_i + \sum\limits_{j=1}^{h_i} (\boldsymbol{k}_i)_j\}$,

$$V = \frac{m}{2\sum\limits_{i=1}^{p}\left(n_i + \sum\limits_{j=1}^{h_i}(\boldsymbol{k}_i)_j\right)} \equiv \frac{m}{2(|\boldsymbol{n}|+|\boldsymbol{K}|)}$$

where $\boldsymbol{K}$ *is the concatenation of all the* $\boldsymbol{k}_i$ *vectors for* $i = 1, \dots, p$*, with* $K_{max} = \max \boldsymbol{K}$ *and* $\boldsymbol{n} = (n_1, \dots, n_p)$*.*

Proof. Under decomposition with the Hamiltonian bound for the general star graphs, as $m \leq 2 \min\limits_{i=1,\dots,p}\{n_i + \sum\limits_{j=1}^{h_i}(\boldsymbol{k}_i)_j\}$

$$V \geq \frac{1}{\sum\limits_{i=1}^{k}\frac{2(n_i+|\boldsymbol{k}_i|)}{m}} = \frac{m}{2\sum\limits_{i=1}^{k}(n_i+|\boldsymbol{k}_i|)}$$

Now under simplification of the centre's, i.e $\bigodot\limits_{i=1}^{p} S_{n_i}^{\boldsymbol{k}_i}$ to $S_{|\boldsymbol{n}|}^{\boldsymbol{K}}$, as $m \geq 2(K_{\max}+1)$

$$V \leq \frac{m}{2(|\boldsymbol{n}| + |\boldsymbol{K}|)}$$

So we have a tight upper and lower bound. □

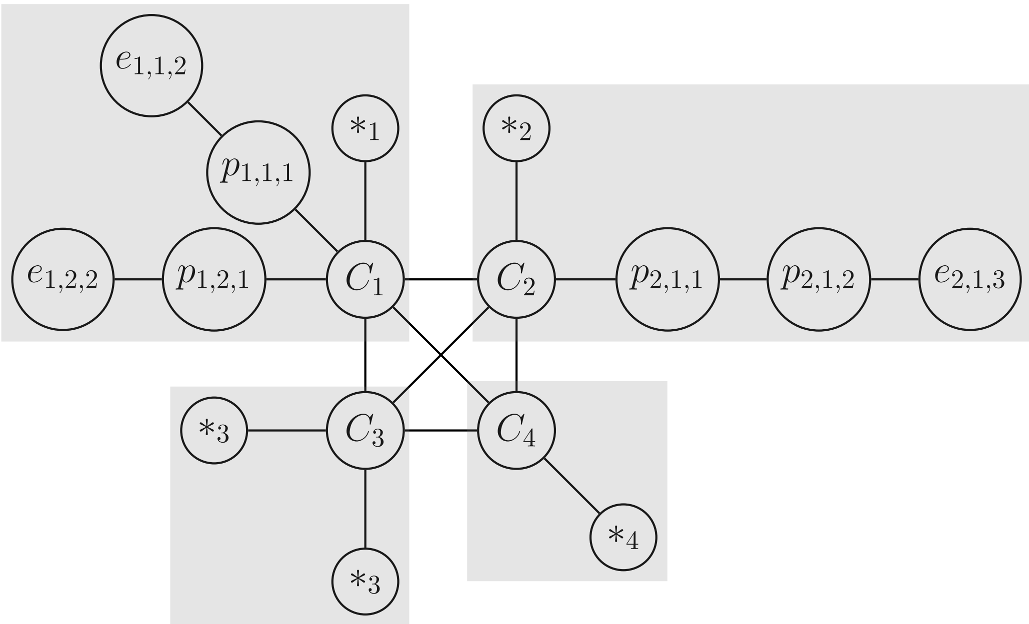


Figure 2.12: Example of decomposition on $(S_3^{1,1}, S_2^2, S_2, S_1)$

Note. We note that this is not the Hamiltonian bound for such a graph and the Hamiltonian bound would be,

$$V = \frac{m}{2(|\boldsymbol{n}| + |\boldsymbol{K}|) + p}$$

Note. The idea of requiring the complete graph of connection between centres is not needed for the separable solution.

3 Strategic Patroller with Random Attackers

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

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

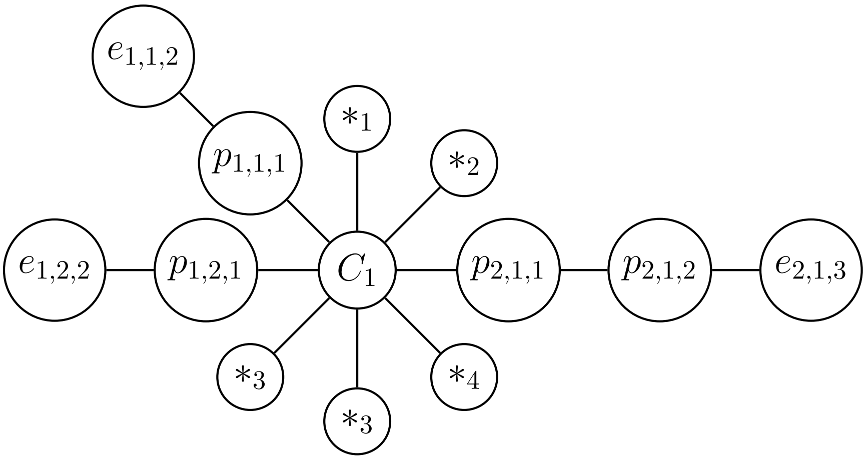


Figure 2.13: Example of simplification on $(S_3^{1,1}, S_2^2, S_2, S_1)$ to $S_8^{2,1,1}$

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

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

and the index reflects this, only wanting the patroller to visit in state, $s = B$, initially and in $s = B + 1$ for a higher cost.

$$W_j(\boldsymbol{s}) = \begin{cases} c_j \lambda_j R_j \text{ for } s_j = B_j \\ c_j \lambda_j \text{ for } s_j = B_j + 1 \\ 0 \text{ Otherwise} \end{cases}$$

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. Looking at the results of only using deterministic attack times on K_6 in a numerical experiment (see Figure **??** we can see a larger amount of errors.

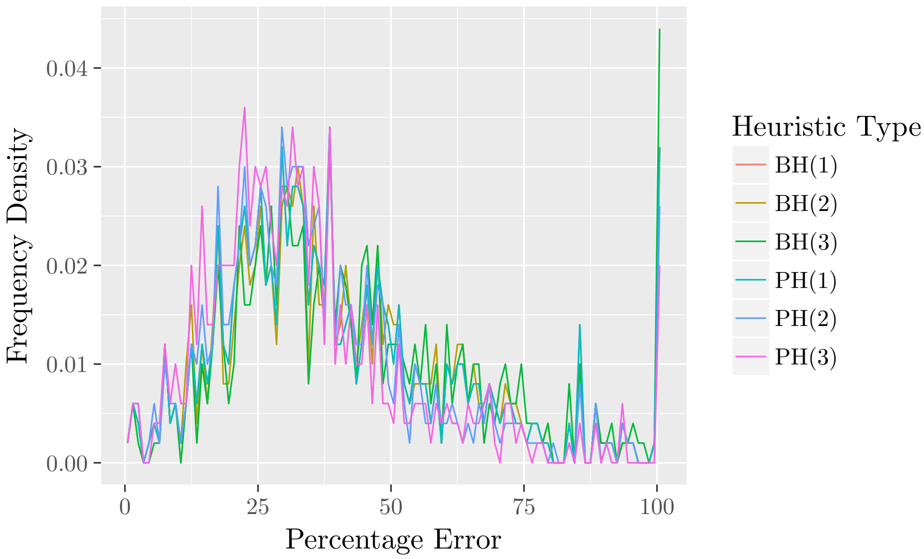


Figure 3.1: Frequency of percentage error with only deterministic attack times

3.2 Proposed Solutions

To 'smooth' the spike in the cost and index, we could suggest splitting up the index in some fashion. Before we suggest splitting the index through we must take into consideration how the benefit and penalty heuristics differ. As the penalty is measuring how much index is not claimed, it should be easy to split it amongst all states $s \leq B$, so that the sum is the correct index. But the benefit is measuring how much index is claimed, so it needs to be cumulative and is therefore a little harder to split amongst all the states $s \leq B$, we do not need that sum is the correct index.

We will first consider splitting the index equally, so for the penalty we use the index

$$W_j^{EP}(\boldsymbol{s}) = \begin{cases} \frac{c_j \lambda R_j}{B} \text{ for } s_j \leq B_j \\ c_j \lambda_j \text{ for } s_j = B_j + 1 \end{cases}$$

and for the benefit use the cumulative version

$$W_j^{EB}(\boldsymbol{s}) = \begin{cases} \frac{s_j (c_j \lambda R_j)}{B} \text{ for } s_j \leq B_j \\ c_j \lambda_j \text{ for } s_j = B_j + 1 \end{cases}$$

We can also consider splitting the index unequally, using the splitting function, on node j, $l_j(x)$ with $x \in [1, ..., B]$ s.t $L_j(B_j) = B_j$ where $L_j(x) = \sum_{u=1}^{x} l_j(u)$, and to place more emphasis on the spike we will require that $l_j(x)$ is non-decreasing. For now we will assume the same splitting is used on each node, dropping the subscript.

$$W_j^{UEP}(\boldsymbol{s}) = \begin{cases} l(s_j) \times \frac{c_j \lambda R_j}{B} \text{ for } s_j \leq B_j \\ c_j \lambda_j \text{ for } s_j = B_j + 1 \end{cases}$$

and for the benefit use the cumulative version

$$W_j^{UEB}(\boldsymbol{s}) = \begin{cases} L(s_j) \times \frac{(c_j \lambda R_j)}{B} \text{ for } s_j \leq B_j \\ c_j \lambda_j \text{ for } s_j = B_j + 1 \end{cases}$$

So that original has $l(x) = 0 \ \forall x \neq B$ and $l(B) = B$. The equal splitting has,$l(x) = 1$. For our numerical study, we will consider using a linear increase $l(x) = \frac{2s_j}{B+1}$.

We now present, at depth 3, both the equal and unequal splitting against its counterpart.

Figure 3.2: Equal and unequal splitting against original for penalty heuristic

Figure 3.3: Equal and unequal splitting against original for benefit heuristic

4 Strategic Patroller with Random Attackers and Local-observations

We now look at altering the work summarized in Section **??** 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

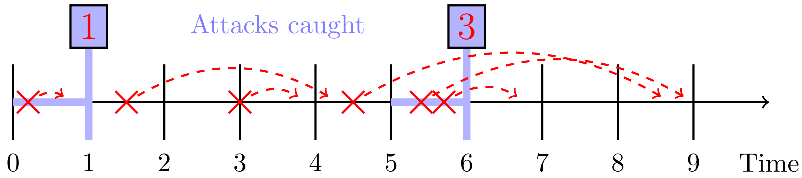


Figure 4.1: Example of timing

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.

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

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.

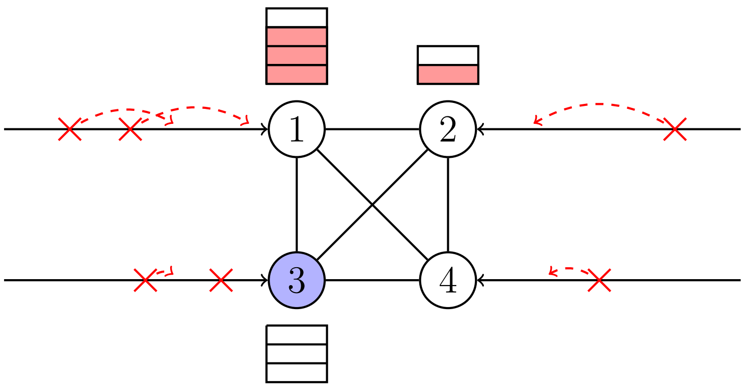


Figure 1: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

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

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.

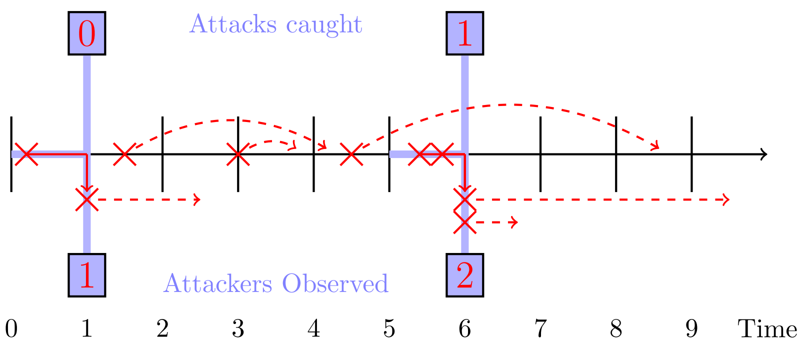


Figure 4.2: Example of timing

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 $l(\boldsymbol{s}, \boldsymbol{v}) = \operatorname{argmin}_i s_i$ 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.

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

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.

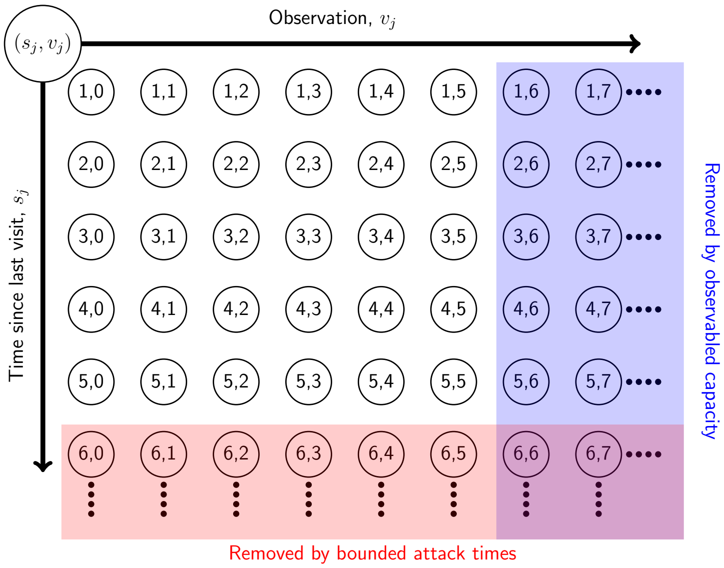


Figure 2: State space diagram for node j, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

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 E.

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

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

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 $s = B$.

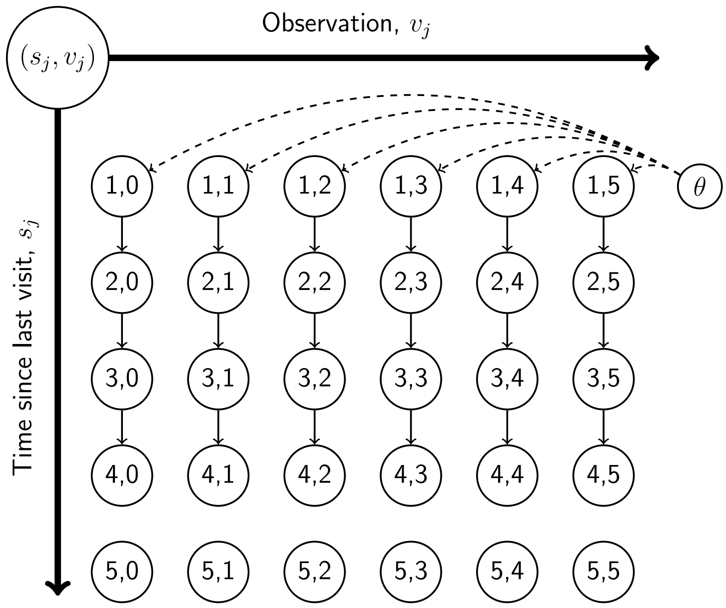


Figure 4.3: Optimal movements for $s < B$

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

$$V_n^{\pi_k}(s, v) = kc\lambda + \omega + E[V_{n-k-1}^{\sigma}(\theta)] \tag{4}$$

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 $s = B + 1$.

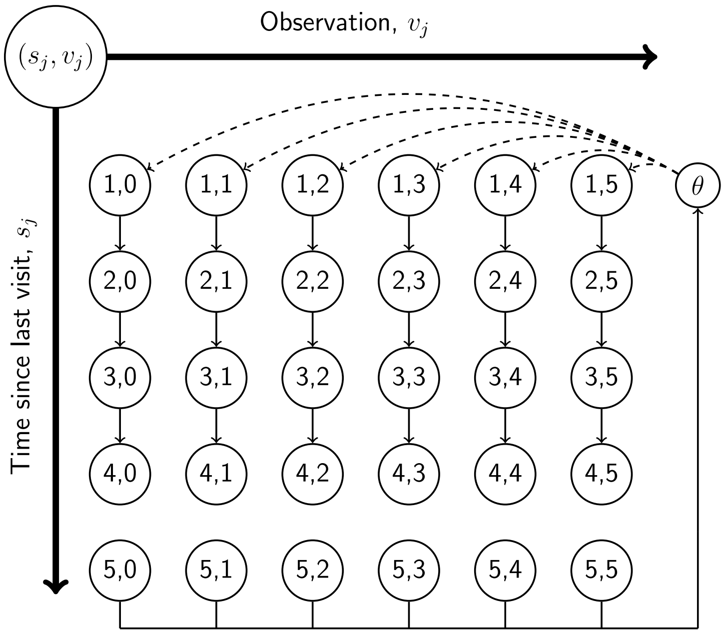


Figure 4.4: Optimal movements for $s < B$ and $s = B + 1$

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

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

We will pick policy, π^{R} over π^{NR} (or be indifferent) if

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

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

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

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 $v_{\mathrm{crit}} \neq 0, v_{\max}+1$

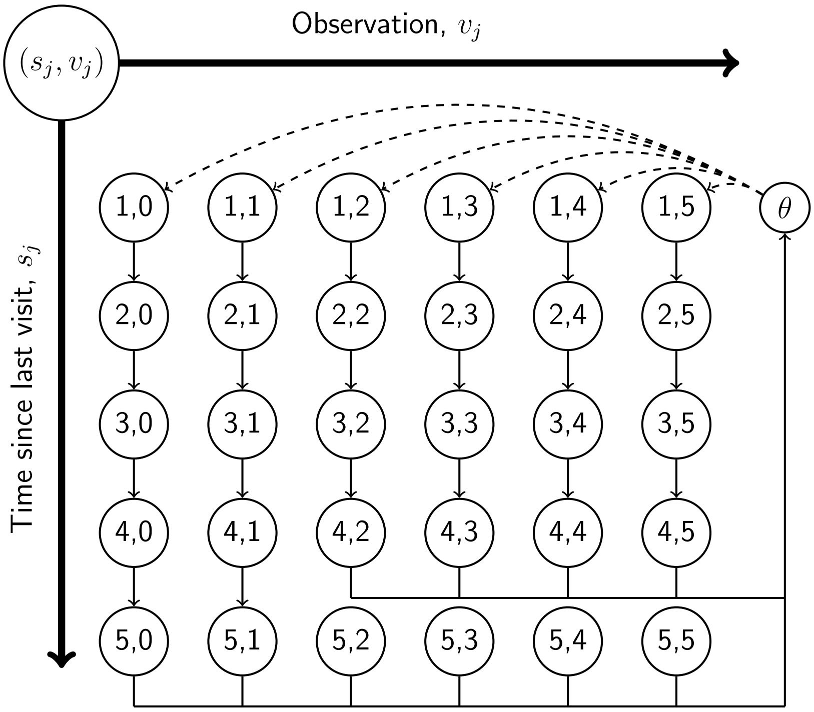


Figure 4.5: Threshold policy, $\pi_{\text{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

- $c\lambda R + cv_{\max} < g \leq c\lambda$ gives $v_{\text{crit}} = v_{\max} + 1$

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 $(\boldsymbol{B}+\boldsymbol{1},\boldsymbol{v})$ 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

$$W_i(s_i, v_i) = \begin{cases} 0 \text{ If } s_i < B_i \\ \Delta_i(v_i) \text{ If } s_i = B_i \\ \Delta_i(v_i + 1) \text{ If } s_i = B_i + 1, v_i < v_{i,\max} \\ \widetilde{\Delta_i} \text{ If } s_i = B_i + 1, v_i \geq v_{i,\max} \end{cases} \tag{5}$$

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 [11].

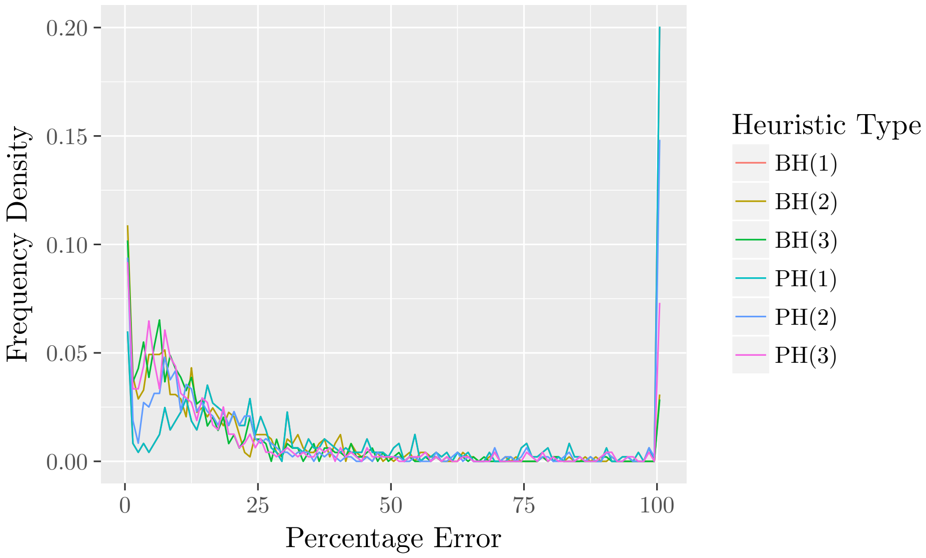


Figure 4.6: Frequency density of percentage errors made by heuristics in simulations

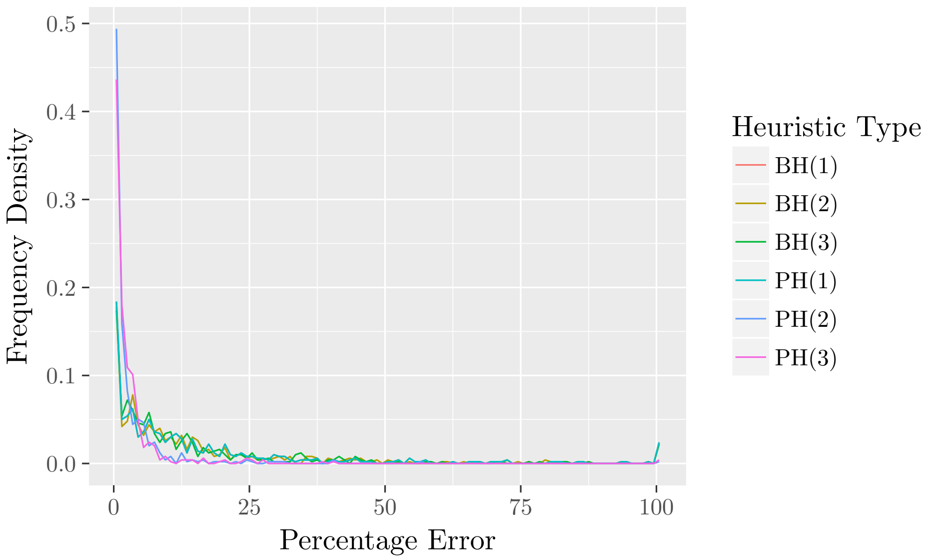


Figure 4.7: Frequency density of percentage errors made by heuristics in simulations for new index

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, $\pi^{\text{TH}}(\boldsymbol{v}_{\text{crit}})$ where $(\boldsymbol{v}_{\text{crit}})_i$ 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 $\boldsymbol{v}_{\text{crit}}$ is a non-increasing 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. We can do the same process in these states, either choosing to wait 0 or 1 time period and then renew, which are the policies π^{R} and π^{NR} respectively. Then

$$
\begin{aligned}
&V_n^{\pi_{\text{R}}}(s,v) = \omega + E[V_{n-1}^{\sigma}(\theta)] \\
&V_n^{\pi_{\text{NR}}}(s,v) = c(vP(B-1 < X \leq B) + \lambda \int_{B-1}^{B} P(X \leq t)dt)\omega + E[V_{n-1}^{\sigma}(\theta)]
\end{aligned}
$$

And the policy π^{R} is picked over π^{NR} (or be indifferent) if

$$
\begin{aligned}
&\lim_{n\to\infty} V_n^{\pi_{NR}}(B,v) - V_n^{\pi_R}(B,v) \geq 0 \\
\iff &\lim_{n\to\infty} c(\lambda \int_{B-1}^{B} P(X \leq t)dt + vP(B-1 < X \leq B)) + E[V_{n-1}^{\sigma}(B+1,0)] - (\omega + E[V_{n-1}^{\sigma}(\theta)]) \geq 0 \\
\iff &g \leq c(\lambda \int_{B-1}^{B} P(X \leq t)dt + vP(B-1 < X \leq B)) = c(\lambda R_B + vp_B)
\end{aligned}
$$

Where $R_x \equiv \int_{x-1}^{x} P(X \leq t)dt = \int_{x-1}^{x} F(t)dt$ and $p_x \equiv P(x-1 < X \leq x) = F(x) - F(x-1)$ (For our generic distributions c.d.f F)

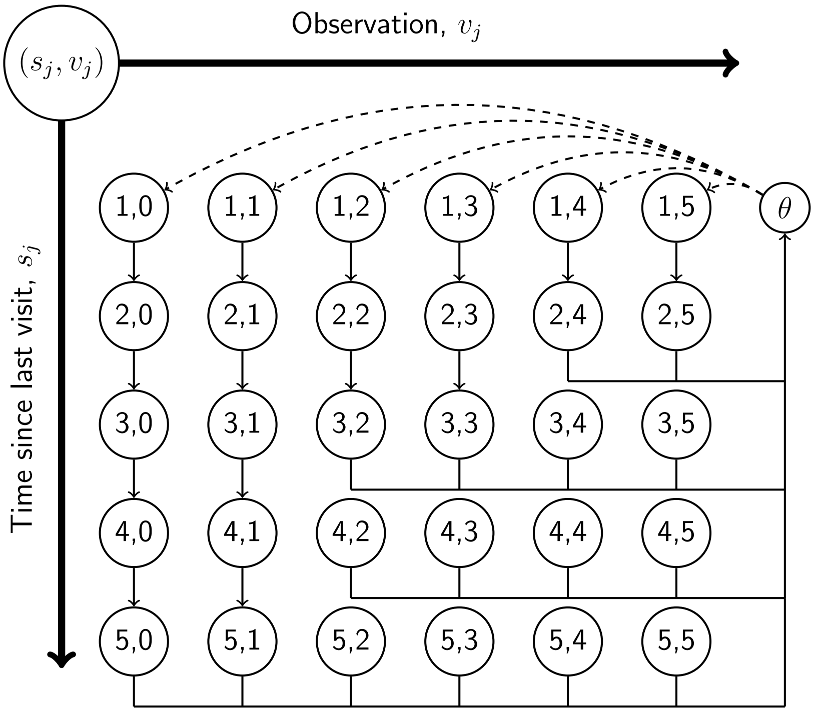


Figure 5.1: Threshold policy, $\pi_{\text{Th}}(6, 4, 2, 2, 0)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

This means the bound for renewing now is almost the same as before, $g \leq c(\lambda R_B + vp_B) \equiv g_B$.

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 $g \leq g_B$ (so if we wait we renew in $s = B$) or otherwise if we wait, we wait again and renew in $s = B+1$.

5.2.1 Plan

Time Estimate: 3 Months

Plan: To complete the backwards recursion case by cases to get the optimal policy to be that of a multi-stage threshold policy. Then to develop an index using the fair price for renewal in each state to allow us to form an index. Implement the index for numerical studies with some simple distributions e.g. uniform and triangular. We also want to investigate whether on two nodes that the IH performs optimally. We want to extend the theory further to have attackers wait until the patroller leaves before beginning their attack.

5.3 Strategic Patroller with Random Attackers on Edges

We want to look at considering similar work to that in section, **??** 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

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

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 $B_{i,j} \equiv \min\{k \,|\, k \in \mathbb{Z}^+ P(X_{i,j} \leq k) = 1\}$ 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 $\widetilde{s}_{i,j} = \min\{s_{i,j}+1, B_{i,j}+1\} \; \forall (i, j) \in E \setminus \{(l(\boldsymbol{s}), j')\}$.

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

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

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 **??** 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 **??**.

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

[1] Steve Alpern and Shmuel Gal. *The Theory of Search Games and Rendezvous*. International Series in Operations Research & Management Science. Kluwer Academic Publishers, Boston, 2003.

[2] Steve Alpern, Thomas Lidbetter, Alec Morton, and Katerina Papadaki. Patrolling a pipeline Book section Patrolling a Pipeline. *Proceedings. Lecture Notes in Computer Science*, 9996:129–138, 2016.

[3] Steve Alpern, Alec Morton, and Katerina Papadaki. Patrolling Games. *Operations Research*, 59(5):1246–1257, 2011.

[4] J. R. Birge and S. M. Pollock. Modelling Rural Police Patrol. *Journal of the Operational Research Society*, 40(1):41–54, 1989.

[5] Gerald Brown, Matthew Carlyle, Javier Salmerón, and Kevin Wood. Defending critical infrastructure. *Interfaces*, 36(6):530–544, 2006.

[6] Jan M Chaiken and Peter Dormont. a Patrol Car Allocation Model: Background. *Management Science*, 24(12):1280–1290, 1978.

[7] Kenneth Chelst. an Algorithm for Deploying a Crime Directed (Tactical) Patrol Force. *Management Science*, 24(12):1314–1327, 1978.

[8] Gustav Feichtinger. A Differential Games Solution to a Model of Competition Between a Thief and the Police. *Management Science*, 29(6):686–699, 1983.

[9] Craig R. Fox, David Bardolet, and Daniel Lieb. Partition Dependence in Decision Analysis, Resource Allocation, and Consumer Choice. *Experimental business research*, XX(19):229–251, 2005.

[10] Richard C Larson. *Urban police patrol analysis*, volume 28. MIT Press Cambridge, MA, 1972.

[11] 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.

[12] Kyle Y. Lin, Michael P. Atkinson, and Kevin D. Glazebrook. Optimal patrol to uncover threats in time when detection is imperfect. *Naval Research Logistics*, 61(8):557–576, 2014.

[13] Roy Lindelauf, Peter Borm, and Herbert Hamers. The influence of secrecy on the communication structure of covert networks. *Social Networks*, 31(2):126–137, 2009.

[14] Katerina Papadaki, Steve Alpern, Thomas Lidbetter, and Alec Morton. Patrolling a Border. *Operations Research*, 64(6):1256–1269, 2016.

[15] Praveen Paruchuri, Jp Pearce, and Janusz Marecki. Efficient Algorithms to Solve Bayesian Stackelberg Games for Security Applications. In *Aaai*, volume 1, pages 2–5, New York, New York, USA, 2008. ACM Press.

[16] Martin L Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons Inc., 1994.

[17] Bernard W Taylor, Laurence J Moore, Edward R Clayton, K Roscoe Davis, and Terry R Rakes. An Integer Nonlinear Goal Programming Model for the Deployment of State Highway Patrol Units. *Management Science*, 31(11):1335–1347, 1985.

Appendices

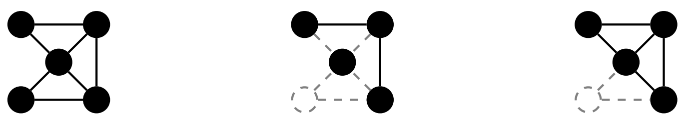
A Graph Definitions

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 (Adjacency matrix)**.** A graph, G, can be reprsented as an *adjacency matrix*, A, where $A_{i,j} = 1 \iff (i, j) \in E$

Definition A.3 (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.4 (Adjacency,Incidency)**.** We say nodes, n, m are *adjacent* if $\exists(n, m) \in E$ and we say the edge (n, m) is *incident* to both n and m.

Definition A.5 (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.6 (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.7. For the graph Q as in Figure 3:

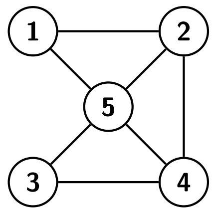


Figure 3: Graph, Q

- An example of a walk is $(1, 2, 1, 5, 4, 2)$
- An example of a trail is $(1, 2, 5, 3, 4, 5, 1)$
- 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.8 (Complete graphs)**.** The *complete graph*, K_n, is a graph of n nodes, in which all edges are present, i.e $e_{i,i'} \in E \, \forall i, i' \in N$.

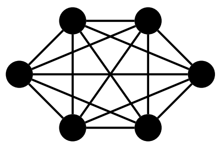


Figure A.2: The complete graph of 6 nodes,K_6.

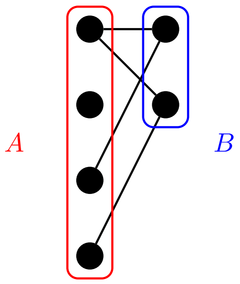
Definition A.9 (Bipartite)**.** A graph is said to be *bipartite* if $N = A \cup B$, where $A \cap B = \emptyset$,and $e_{i,i'} \notin E \, \forall i, i' \in A$, $e_{i,i'} \notin E \, \forall i, i' \in B$.

Definition A.10 (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 $e_{i,i'} \in E \, \forall i \in B \, \forall i' \in A$.

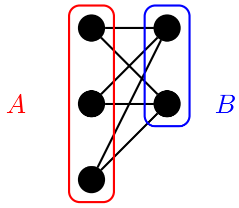
Definition A.11 (Subdivision,Smoothing)**.** A *Subdivision* (or *expansion*) of a graph, G, is a new graph G' which is made by subdividing a chosen edge. The subdivision of an edge $\{u, v\}$ yields a graph with a new node w and the splitting of the edge $\{u, v\}$ into $\{u, w\}$ and $\{w, v\}$.

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.

Definition A.12 (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.



(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.

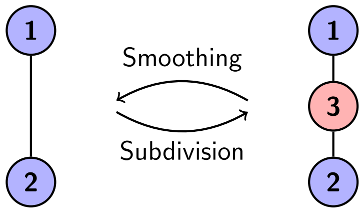


Figure A.4: Subdivision and Smoothing of the edge $\{1, 2\}$

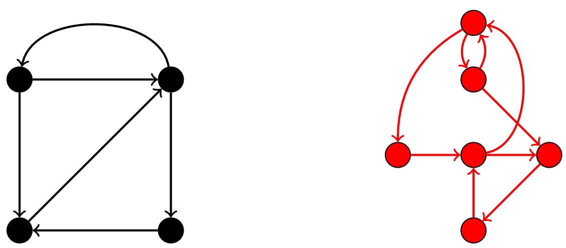


Figure A.5: A graph G and its directed edge-vertex dual, $EV(G)$

B Strategic Patroller and Strategic Attacker

B.1 Proof of diametric waiting time

Consider visit i to a end node capturing C_i of the attacks placed by the diametric attack, then the total number of attacks captured is $C = \sum_{i=0}^{\lfloor \frac{T-1}{\bar{d}} \rfloor} C_i$.

Then leaving the initial node at time t gives us, $C_0 = \min(t, T-m+1)$, $C_i = \big(\min(t+i\bar{d}, m, T-(t+i\bar{d})\big)_+$ for $i \neq 0$.

We first note that C_0 is increasing in the region $t \leq T-m$ and constant for $t \geq T-m+1$. So C_0 is concave. Due to this being increasing, and therefore its only possible to have others increasing if $t \leq T-m$, we will make this an assumption.

Now we look at C_i and see it is increasing for the region $t \leq m-1-i\bar{d}$, constant for $m-i\bar{d} \leq t \leq T-1-m-i\bar{d}$ and decreasing for $T-m-i\bar{d} \leq t \leq T-i\bar{d}-1$. So C_i is concave. Hence C is concave and so finding the best choice for t is when the net increase is constant (or decreasing for the first time).

We see that C_0 always improves, contributing 1 to the net increase. C_i however is only increasing, and contributes 1, if $i \leq \frac{m-1-t}{\bar{d}} < 2$ (as $m \leq 2\bar{d}$), so only C_1 can possibly contribute an increase when $t \leq m-\bar{d}-1$.

C_i is decreasing, contributing -1 if $\frac{T-m-t}{\bar{d}} \leq i \leq \frac{T-1-t}{\bar{d}}$, with at most 2 C_i's being decreased (as the gap is $\frac{m}{2\bar{d}}$ and $\bar{d} < m \leq 2\bar{d}$).

This worst issue occurs when $\frac{T-m-t}{\bar{d}}$ or $\frac{T-1-t}{\bar{d}}$ are integers, meaning we have chosen $t = T-m-k\bar{d}$ or $t = T-1-k\bar{d}$ for some $k \in \mathbb{Z}$.

So overall increasing t to $t+1$, with $t \leq m-\bar{d}-1$, gives us a net increase 1 when we have non-integers, 0 when we have integers and -1 if $t > m-\bar{d}-1$.

So we pick the upper concave region, as its about to go from increasing to decreasing, giving us a choice of $t = m-\bar{d}$.

Note. $t = m-\bar{d}-1$ is not net decreasing, but $t = m-\bar{d}-1$ is net decreasing, so $t = m-\bar{d}$ is a choice for the maximum.

B.2 Proof of conditions on T for diametric attack

B.3 Proof of time-limited diametric attack

B.4 Generalised ARHP

C Improving random oscillations

C.1 Reason For Probability of Interception formula

To argue why this is consider looking at just the integer points on the graph first, then by looking whether it is best to return via the "left return" or a "right return", that is the shortest time to the next return along the oscillation. The time/distance to return via the left motion, i.e through 1, is $2(i-1)$ and the time/distance to return via the right motion, i.e through c and the set of $*$'s, is $2(n+k-(i-1)) = 2(n+k+1-i)$. So the categories fall when $2(i-1) < 2(n+k+i-1), 2(i-1) = 2(n+k+1-i)$, and $2(i-1) > 2(n+k+1-i)$. These decide the boundaries and they return with these distances.

Therefore the patroller catches the m initially and then either another t or m attacks on the return, where t is the time to next return (either $2(i-1)$ or $2(n+k+1-i)$). This means that the patroller catches $\min(m+t, 2m)$ attacks and hence the probability of interception is $\frac{\min(m+t,2m)}{2(n+k)}$ as they cycle is of length $2(n+k)$ and the number lies in $\min(m+t, 2m)$ of these.

For c consider that it catches m attacks initially and is returned to $n-1$ times each at a distance/time of 2 apart, meaning it gains $t = 2$ or m attacks $n-1$ attacks. Hence it intercepts $\min(m+2(n-1), nm)$ out the cycle length of $2(n+k)$.

Finally $*$ nodes are easy as they are they are not returned to in a single cycle, hence they intersect m out the possible $2(n+k)$.

C.2 Naive Improvement Complete analysis

1. If $M = \emptyset, R = \emptyset$ then $C_{min}(\alpha) = \min\{C^L_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, q_S\}$. As the Patroller wishes to select q_L, q_S to maximize this probability, the problem becomes

$$\begin{array}{rc} \text{Maximize} & (1-q_L-(n-1)q_S)\frac{m}{2(n+k)} + \min\{q_L, q_S\} \\ \text{Subject to} & q_L + (n-1)q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

   Meaning that as due to the symmetry of q_L, q_S we must have that $q_L = q_S$ which means improvement is only possible if $2(n+k) - nm \geq 0$, then set $q_L = q_S = \frac{1}{n}$. Meaning $P = 0, Q_L = \frac{1}{n}, Q_R = \frac{n-1}{n}$ and giving a lower bound $V \geq \frac{1}{n}$.

2. If $M \neq \emptyset, R = \emptyset$ then $C_{min}(\alpha) = \min\{C^L_{min}, C^M_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} + 2 \rfloor), Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, P\frac{m}{2(n+k)}, q_S\}$. As the Patroller wishes to select q_L, q_S to maximize this probability, the problem becomes

$$\begin{array}{rc} \text{Maximize} & (1 - q_L - q_S)\frac{m}{2(n+k)} + \min\{q_L, (1 - q_L - (n-1)q_S)\frac{m}{2(n+k)}, q_S\} \\ \text{Subject to} & q_L + (n-1)q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that as due to the symmetry of q_L, q_S we must have that $q_L = q_S$, with improvement only possible if $2(n+k) - nm \geq 0$, which means we seek to maximize $(1 - nq_L)\frac{m}{2(n+k)} + \min\{q_L, (1 - nq_L)\frac{m}{2(n+k)}\}$ giving $q_L = \frac{m}{2(n+k)+nm}$ (as one is then decreasing in q_L and one is increasing in q_L). Meaning $P = \frac{2(n+k)}{2(n+k)+mn}, Q_L = \frac{m}{2(n+k)+mn}, Q_S = \frac{m(n-1)}{2(n+k)+mn}$ and giving a lower bound $V \geq \frac{2m}{2(n+k)+mn}$

3. If $M \neq \emptyset, R \neq \emptyset$ then $C_{min}(\alpha) = \min\{C^L_{min}, C^M_{min}, C^R_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} + 2 \rfloor), Pw(k+1) + q_S, Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, P\frac{m}{2(n+k)}, q_S\}$. Now was $Pw(k+1) + q_S \geq PW(*) + q_S$, we can ignore this element (meaning here it does not matter if R is empty or not). As the Patroller wishes to select q_L, q_S to maximize this probability, the problem becomes

$$\begin{array}{rc} \text{Maximize} & (1 - q_L - (n-1)q_S)\frac{m}{2(n+k)} + \min\{q_L, (1 - q_L - (n-1)q_S)\frac{m}{2(n+k)}, q_S\} \\ \text{Subject to} & q_L + (n-1)q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that as due to the symmetry of q_L, q_S we must have that $q_L = q_S$, with improvement only possible if $2(n+k) - nm \geq 0$, which means we seek to maximize $(1 - nq_L)\frac{m}{2(n+k)} + \min\{q_L, (1 - nq_L)\frac{m}{2(n+k)}\}$ giving $q_L = \frac{m}{2(n+k)+nm}$ (as one is then decreasing in q_L and one is increasing in q_L). Meaning $P = \frac{2(n+k)}{2(n+k)+mn}, Q_L = \frac{m}{2(n+k)+mn}, Q_S = \frac{m(n-1)}{2(n+k)+mn}$ and giving a lower bound $V \geq \frac{2m}{2(n+k)+mn}$

C.3 Combinatorial Improvement analysis

1. If $M = \emptyset, R = \emptyset$ then $C_{min}(\beta_2) = \min\{C^L_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, q_S\}$. As the patroller wishes to select q_L, q_S to maximize this probability the problem becomes

$$\begin{array}{rc} \text{Maximize} & (1 - q_L - \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_S)\frac{m}{2(n+k)} + \min\{q_L, q_S\} \\ \text{Subject to} & q_L + \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that due to the symmetry of q_L, q_S we must have that $q_L = q_S$, which means that improvement is only possible if $2(n+k) - m(1 +$

$\frac{n-1}{\lfloor \frac{m}{2} \rfloor}) \geq 0$. Then setting $q_L = q_S = \frac{\lfloor \frac{m}{2} \rfloor}{\lfloor \frac{m}{2} \rfloor + n-1}$. Meaning $P = 0, Q_L = \frac{\lfloor \frac{m}{2} \rfloor}{\lfloor \frac{m}{2} \rfloor + n-1}, Q_S = \frac{n-1}{\lfloor \frac{m}{2} \rfloor + n-1}$ and giving a lower bound of $V \geq \frac{\lfloor \frac{m}{2} \rfloor}{\lfloor \frac{m}{2} \rfloor + n-1}$

2. If $M \neq \emptyset, R = \emptyset$ then $C_{min}(\beta_2) = \min\{C^L_{min}, C^M_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} \rfloor + 2), Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, P\frac{m}{2(n+k), q_S}$. As the patroller wishes to select q_L, q_S to maximize this probability the problem becomes

$$\begin{array}{ll} \text{Maximize} & (1 - q_L - \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_S)\frac{m}{2(n+k)} + \min\{q_L, (1 - q_L - \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_S)\frac{m}{2(n+k)}, q_S\} \\ \text{Subject to} & q_L + \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that due to symmetry of q_L, q_S we must have that $q_L = q_S$, which means that the improvement is only possible if $2(n+k) - m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor}) \geq 0$. Then setting $q_L = q_S$ means we must maximize $(1 - (1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_L)\frac{m}{2(n+k)} + \min\{q_L, 1 - (1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor} q_L)\frac{m}{2(n+k)}\}$ giving $q_L = \frac{m}{2(n+k) + m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor}}$ (as one is increasing in q_L and one is decreasing in q_L). Meaning $P = \frac{2(n+k)}{2(n+k) + m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor})}, Q_L = \frac{m}{2(n+k) + m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor})}, Q_S = \frac{m(n-1)}{\lfloor \frac{m}{2} \rfloor (2(n+k) + m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor}))}$ and giving a lower bound of $V \geq \frac{2m}{2(n+k) + m(1 + \frac{n-1}{\lfloor \frac{m}{2} \rfloor})}$

3. If $M \neq \emptyset, R \neq \emptyset$ then $C_{min}(\beta_1) = \min\{C^L_{min}, C^M_{min}, C^R_{min}, C^S_{min}\} = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} \rfloor + 2), Pw(k+1) + q_S, Pw(*) + q_S\} = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} \rfloor + 2), Pw(*) + q_S\} = P\frac{m}{2(n+k)} + \min\{q_L, P\frac{m}{2(n+k)}, q_S\}$ As $Pw(k+1) + q_S \geq Pw(1) + q_S$, we can ignore the element (meaning here it does not matter if R is empty or not).As the patroller wishes to select q_L, q_S to maximize this probability the problem becomes

$$\begin{array}{ll} \text{Maximize} & (1 - q_L - q_S)\frac{m}{2(n+k)} + \min\{q_L, (1 - q_L - q_S)\frac{m}{2(n+k)}, q_S\} \\ \text{Subject to} & q_L + q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that due to symmetry of q_L, q_S we must have that $q_L = q_S$, which means that the improvement is only possible if $n + k - m \geq 0$. Then setting $q_L = q_S$ means we must maximize $(1 - 2q_L)\frac{m}{2(n+k)} + \min\{q_L, (1 - 2q_L)\frac{m}{2(n+k)}\}$ giving $q_L = \frac{m}{2(n+k+m)}$ (as one is increasing in q_L and one is decreasing in q_S). Meaning $P = \frac{2(n+k)}{2(n+k+m)}, Q_L = \frac{m}{2(n+k+m)}, Q_S = \frac{m}{2(n+k+m)}$ and giving a lower bound of $V \geq \frac{2m}{2(n+k+m)}$.

C.4 Combinatorial improvement extension

We are only dealing with m odd so $m' = m + 1$ even and $M \neq \emptyset$ $R = \emptyset$. We will play the hamiltonian bound with probability P, the left end-ensuring improvement with probability q_L and the a particular right loop with probability q_S (so $Q_S = \frac{n-1}{\frac{m'}{2}} q_S$.

$C_{min}(C^L_{min}, C^M_{min}, C^S_{min}) = \min\{Pw(1) + q_L, Pw(\lfloor \frac{m}{2} \rfloor + 2)Pw(*) + \frac{m}{m'} q_s\} = P\frac{m}{2(n+k)} + \min\{q_L, P\frac{m}{2(n+k)}, \frac{m}{m'} q_S\}$ As the patroller wishes to select q_L, q_S to maximise the probability the problem becomes

$$\begin{array}{rc} \text{Maximize} & (1 - q_L - \frac{n-1}{\frac{m'}{2}} q_S)\frac{m}{2(n+k)} + \min\{q_L, (1 - q_L - \frac{n-1}{\frac{m'}{2}} q_S)\frac{m}{2(n+k)}, \frac{m}{m'} q_S\} \\ \text{Subject to} & q_L + \frac{n-1}{\frac{m'}{2}} q_S \leq 1 \text{ (Probability sum constraint)} \\ & q_L, q_S \geq 0 \end{array}$$

Meaning that we must have $q_L = \frac{m}{m'} q_S$ which means the improvement is only possible if $m + 2(n-1) \leq 2(n+k)$ (for similarity to even normal combinatorial improvement $\iff m(1 + \frac{n-1}{\frac{m}{2}}) \leq 2(n+k)$).

Then setting $q_L = \frac{m}{m'} q_S$ means we must maximize $(1 - \frac{m+2(n-1)}{m} q_L)\frac{m}{2(n+k)} + \min\{q_L, (1 - \frac{m+2(n-1)q_L}{m})\frac{m}{2(n+k)}\}$ giving $q_L = \frac{m}{2(n+k)+m+2(n-1)}$. Meaning $P = \frac{2(n+k)}{2(n+k)+m+2(n-1)}$, $Q_L = q_L = \frac{m}{2(n+k)+m+2(n-1)}$ and $Q_S = \frac{n-1}{\frac{m'}{2}} q_S = \frac{2(n-1)}{m} q_L = \frac{2(n-1)}{2(n+k)+m+2(n-1)}$. Giving a lower bound of $V \geq \frac{2m}{2(n+k)+m+2(n-1)}$

Note. It is worth noting that this is analogous to the result found by the usual combinatorial improvement when m is even.

D Optimal Solution for a Random Attacker

E Optimal Solution for Random Attacker with Local-Observations