Second Year Report

Thomas Lowbridge
School of Mathematical Sciences
University of Nottingham

June 22, 2018

Contents

Summary of report

In section 1 we provide an overview of the current work done on security problems, which focuses on a patroller attempting to stop some malicious entity. We will also provide an in-depth review of [7] and [18] upon which our extensions in sections 2 and 4 will be based.

In section 2 we continue last year's initial findings and provide new optimal strategies for the patroller, as well as correcting an optimal attack strategy which solves a ambiguity with Lemma 9 in [7]. We further develop the idea the in order to provide us with tools to handle a wider class of graphs.

In section 3 we analyse the work done in [7] more closely, particularly looking if their issue with having only deterministic attack times and the idea of having too much knowledge. This is in an aim to see if there is any evidence that the error of our heuristic in our later work(section 4) are exhibited in the original problem's heuristics. We introduce the idea of splitting the index among non-acting states which will be use in our extension.

In section 4 we build on the work done in [18] by introducing the concept of the patroler who can move instantaneously. And later who observes suspicious behaviour of arriving attackers and try to incorporate this information in an effort to develop a heuristic which is near optimal.

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 type of security problems.

We leave most proofs to the Appendix. We also provide some definitions of 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 multiple resources are available, it is a question of where to deploy then and for what length of time, such as a fire brigade being deployed to sectors to fight a spreading forest fire. However 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 ([17],[13],[12]), using many technieques inclduing statistics based on $M/M/c$ queues(with small modifications due to the stochasticity of the number of servers) for the purpose of alloacting patrols, later moving onto patrolling rural areas ([9]), in which the primary difference was the incorporation of the travel time into the service rate. Furthermore work was done on police patrolling highways([28]), . 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.The solution to the problems involved randomizing the patrols in order to maximize the probability of intercepting a crime in progress.

While the work done in scheduling police patrols are a strategic only from the police force's point of view, one may wish to introduce a strategic problem for the attacker.This makes the problem a game-theoretic one, while differential game formulations exist ([14]) 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.

A baseline to start searching for an entity is the search game, a game-theoretic formulation, which involves a searcher aiming to minimize the time until they find either an immobile or mobile hider in some search space ([5]). As a special search space, graphs are considered to just be subset of $\mathbb{R}^3$ (or $\mathbb{R}^2$ if planar) by embedding.

Game-theoretic analyses have been done in studies of counter-terrorism actions ([11],[21]), to find how a patroller should randomize their patrol. It has been shown that the obvious strategy of; given n targets, spend $\frac{1}{n}$ of the available patrol time on every target; is not necessarily a good idea([15]). The game has also been formulated as a Stackleberg (leader-follower decisions) type game, instead of a simultaneous decision one, in which near optimal heuristics have been found([24]).

Many extensions the search game have been studied, such as a searcher who has multiple modes(such as fast or slow), they can search in, this work was applied to a predator prey model, to analyse at what frequency a predator should ambush the prey ([4]). Another variation of the search game, is the accumulation game, in which the hider can hide portions of their self across the search space, with some critical amount of their self which needs to avoid being found in roder to be effective. For a continuous portioning of the hider in a discrete ([16]) and continuous ([26]) search space, the strategies were found be pure. While later this studied with a graphical structure and it was found that mixed strategies need to be employed in optimum([3]).

Another common assumption about the search space is that we can divide it into cells, then the hider is in one of these cells. We assume some initial probability distribution of where the hider is, is known the searcher and that searching a cell costs the searcher a differing amount dependent on the cell. Then the seracher upon completion of a search where the hider is not find, can update the prior probability in a bayesian fashion([10],[22]). It has been found that the searcher has a uniformly optimal strategy for all time horizons and so even if the searcher has an unknown deadline, to find the hider, they can act optimally ([20]). Work has also been done to incorporate the portioning of the hider as in the accumulation game, but when we have search cells([27]) and further when these cells have a limited capacity of how much the hider can hide in them([29],[30])

While search games (and there successors) are of major importance when there are two uncooperative element, they have also been expended to rendezvous search games([5]). A rendevous game is a game where two rendezvours wish to detect each other in the search space. Such games change the min-max objective of search games into a min-min objective. While rendezvous games do not inherintely have a mallicous entity, they are still useful in secruity problems such as, the regrouping of military assets in an unknown environment and with the development of communication technologies have been applied to the frequencies on military assets communicate. Like before often a graphical structure can be applied ([2]). To reintroduce a mallicous entity in the field of communication, a searcher is introduce to create a three player problem , with two rendezvours who are cooperative and a enemy searcher. This idea has been applied to the wire-tapping of phone lines([8]) and the jamming of signals in millitary communications([1]).

A patrolling game ([7]) takes the search game in which the hider is always in the search space, and allows the hider (now called an attacker) to only be in the space for a subset of the time horizon, which is chosen by the attacker. This means that the patrol games are win-lose games instead of games of degree like the original search game. Section 1.2 reviews the formulation and initial results of patrol games on graphs. 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 ([23]). A continuous time version of the patrol game has also been formulated and solved for the line graph, agreeing with the discrete case ([6]). Throughout it is proposed that the intuitive solution, of simply going back and forth on the line, is not

optimal and that it is the size of the gap between visits relative to how long the attacker is in the network that matters. In section 2 we propose an issue with one attacking strategy developed when the time horizon is finite, correcting the issue and proposing a more general application. This is all done with the aim of getting a solution for all tree graphs, which provide a great lower bound for any graph.

Following this work, a infinite time horizon patrol problem was formulated for generic attack times which can differ between nodes ([18]), 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. In section 3 we alter the theory to deal with a problem, where the patroller may observe some suspicious behaviour from attackers and incorporates this information into their decision, but first we look at accounting for instantaneous movement of a patroller in the graph, in order to model that of guard flicking through live-video feeds.

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 ([19])

1.2 Review of Strategic Patrolling games

This section summarizes the key work from [7] and their work on patrolling games. This will provide a baseline for future work done in 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

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

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 [7] 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 group of patrols and attackss 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 $V \leq \frac{\omega}{n} \leq \frac{m}{n}$, where ω is the maximum number of nodes the patroller can visit in the interval, I of length m.

Decomposition(Patroller):

We can consider decomposing the graph into subsets of nodes, and hence (edge-preserving) sub-graphs, Q_i, with known values, $V_i = V(Q_i)$.Then by playing on these sub-graphs with some appropriate probabilities, $p_i = \frac{1}{\sum_{j=1}^{n} \frac{V_i}{V_j}}$, the patroller can achieve $V \leq \frac{1}{\sum_{j=1}^{n} \frac{1}{V_j}}$, with equality if the decomposition is disjoint. An example of this can be found in Appendix B.1

Simplification(Patroller and Attacker):

The 'merging' of adjacent nodes (Formally Node identification) reducing the number of nodes. When this merging is repeated the graph is said to be simplified.

Definition 1.1 (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 $W' = \pi(W)$.

If we embed an optimal walk on Q into Q' (a simplified version of Q), then it is shown that $V(Q') \geq V(Q)$. It is worth noting that from this perspective, the bound on Q' is an attacker bound. However by considering an opposite of

simplification we can get patroller bounds. An example of this can be found in Appendix B.2

Diametric attack(Attacker)

Attacking with equal probability at a pair of points which are the maximum distance apart, starting at attack times $\tau = 0, 1, ..., T-m$ with equal probability is called the *diametrical attack*. intervals(starting at $\tau = 0, 1, ..., T-m$. By using such an attack the diametric bound in Lemma 9 (from [7]) $V \leq \max\left\{\frac{m}{2d}, \frac{1}{2}\right\}$

While this bound can hold, an issue caused by the time-horizon is ambiguous in the statement of the Lemma. We will discuss this ambiguity and provide a more concrete statement which considers the time-horizon.

Covering(Patroller) and Independence(Attacker):

A collection of patrols which each capture every node within there patrol is called *covering* set, with the minimal number of patrols needed called the *covering number*, $\mathcal{C}$. As the patroller can implement one of these patrols, $V \geq \frac{1}{\rfloor}$.

A set of nodes in which any patrol cannot catch more than one is called an *independent* set, with the maximal number of nodes which can be used called the *independence number*, $\mathcal{I}$. As the patroller can play at one of these nodes with equal probability, $V \leq \frac{1}{I}$.

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).

1.2.3 Solved Graphs

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

Hamiltonian:

In a Hamiltonian graph, following a Hamiltonian cycle(called a *Hamiltonian patrol*) means the patroller sees every node every n time periods, so if $m \geq n$ all attacks will be caught, hence it is assumed that $m < n$ (otherwise $V = 1$).

To extend this the *random Hamiltonian patrol* is a patrol strategy starting with equal probability at all nodes on the graph and patrolling by following a fixed Hamiltonian cycle. It is shown that using this strategy gives $V \geq \frac{m}{n}$, as the patroller sees each node, at each time with this probability. Now with a general attacker bound, they get the result that $V = \frac{m}{n}$ for Hamiltonian graphs.

Two classes of such graphs are cyclic graphs, C_n and the complete graphs, K_n and we note that the extra edges in the class of complete graphs do not effect the value of the game.

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, [7] and [23]. 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, and corresponding values, for comparison to solving our more generic class of graphs later.

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

It is worth highlighting the strict concavity of the value function for the line in the region, S_5, as this is the region that will cause us the most issue when trying to generalise their results. See Figure 1.1.

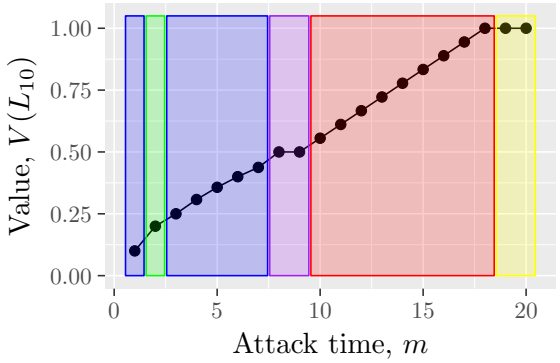


Figure 1.1: Value of game onL_{10} with attention drawn to $(10, m)$ split into regions; S_1,S_2,S_4,S_5,S_3

We also note that S_2 uses the diametric attack in [23], and so its ambiguity in the time horizon can be resolved using the alteration in 2

Bipartite:

In a bipartite graph, with sets disjoin sets A and B (WLOG assume $|B| \geq |A|$),then the *Bipartite Attack* selects nodes will equal probability from B for a fixed time interval, I (or for the two time intervals, I and $I+1$ with equal probability if m is odd). It has been shown that by the patroller following the attack $V \leq \frac{m}{2|B|}$ as the patroller much alternate between the sets, seeing

a potential attack every other time periods(during the attack interval). When $m \geq 2|B|$ this alternating guarantees capture and $V = 1$.

On a complete biparitate, $K_{b,b}$ which is a hamiltonian graph, the patroller bound gets a tight lower bound so they show that $V = \frac{m}{2b}$.

Note. When m is odd, the use of two fixed time intervals I and $I + 1$ is made to 'balance' the fact that it is possible to see $\frac{m+1}{2}$ possible attacks, but with the offset time intervals two of these only see half a potential attack.

1.2.4 Conclusion on review of patrolling problem

While [7] introduced the patrolling game and along with [23] solved the game for some of the most common graphical structures, they present no solution for generic graphs and comment that exact solutions to graphs are difficult to find. However with common graph classes solved they have provided a good baseline for our future work, which will continue their theory, discussing new strategies and their applications to a more generalised line graph, called the extended star graph.

1.3 Review of patrolling problem with random attackers

As discussed towards the end of section 1.2, it proves quite difficult to find exact solutions to the patrolling game for a general graph. With the aim to develop strategies to the patrolling game on any graph, [18] first develops a similar patrolling problem which is only strategic for the patroller but uses stochastic attack time distributions in the infinite time horizon. They manage to find a near optimal heuristic for the patroller. Later they reintroduce the game-theoretic element of a strategic attacker by formulating a normal-form(matrix) game, in which the patroller has only a finite list of strategies, picked to be 'appropriately good'.

1.3.1 Problem set-up

A patrolling problem with random attackers, $G = G(Q, \boldsymbol{X}, \Lambda, \boldsymbol{p}, \boldsymbol{c})$ 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 Poisson process arrival rate, Λ.
- A vector of node choosing probabilities, $\boldsymbol{p} = (p_1, ..., p_n)$
- A vector of costs, $\boldsymbol{c} = (c_1, ..., c_n)$.

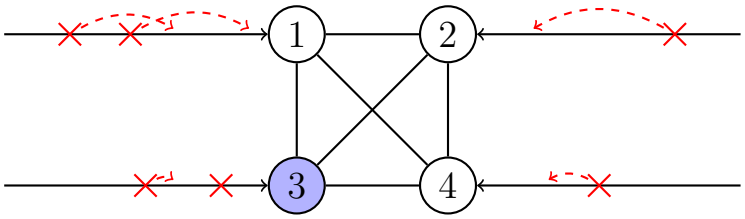


Figure 1.2: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

The attackers arrive to the system at a rate of Λ, at which point they pick to attack node i with probability p_i. They begin their attack immediately, which lasts an amount of time drawn from the nodes attack time distribution, X_i. 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).

By Poisson thinning it is possible to divide the overall Poisson arrival process to individual Poisson arrival processes at each node, with rate $\lambda_i = \Lambda p_i$.

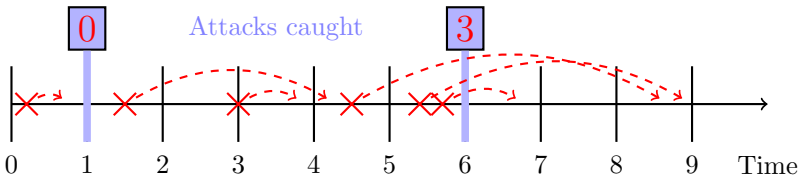


Figure 1.3: Example for a given node, when the patroller visits at times, $1, 5$

We can formulate such as problem as a Markov Decision Process(MDP) with state space, $\Omega = \{\boldsymbol{s} = (s_1, ..., s_n) \,|\, s_i = 1, 2, ... \text{ for } i = 1, ..., n\}$, 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 $l(\boldsymbol{s}) = \text{argmin}_i\, s_i$. The available decisions from state $\boldsymbol{s}$ are $\mathcal{A}(\boldsymbol{s}) = \{j \,|\, A_{l(\boldsymbol{s}),j} = 1\}$ and when node i is chosen by the patroller the state transitions to $\phi(\boldsymbol{s}, i) = \widetilde{\boldsymbol{s}}$, where $\widetilde{s}_i = 1$ and $\widetilde{s}_j = s_j + 1 \; \forall j \neq i$.

Because the future of the process is independent of its past, it is only the current state that matters, 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 $C(\boldsymbol{s}, i) = \sum_{j=1}^{n} C_j(\boldsymbol{s}, i)$, 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

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

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 [25]), 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 [25]). 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 $C^{\mathrm{OPT}}(\boldsymbol{s}_0)$ does not depend on the initial state and so $C^{\mathrm{OPT}} = C^{\mathrm{OPT}}(\boldsymbol{s}_0)$ is the same for all initial states.

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 E. 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

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

For ease of notation we will define $\pi_i : \Omega \to \alpha_i \in \Pi_i^{\mathrm{MN}}$, 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 $C^{\mathrm{OPT}} \geq C^{\mathrm{MN}} \equiv \min_{\pi^n \in \Pi^{\mathrm{MN}}} \sum_{i=1}^{n} V_i(\pi^n)$.

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 1.2 (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.

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 broken 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 by using a look-ahead window of length l we can decide where to move. We look at all paths of length l from the current node, and then aggregate the index along the each path and then choose 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 and by using a look-ahead window of length l we can decide where to move. We look at all paths of length l from the current node, and then aggregate the indices not along the path and then choose 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 $l+1$. 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 [18]. A key result of the study, that we will look at later is that IPH seems to outperform 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 $\bar{d} < m \leq 2\bar{d}$), by considering 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 to Lemma 1.10 consider the following

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 C.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 [7].

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 C.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 C.3

2.2 New Tools

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

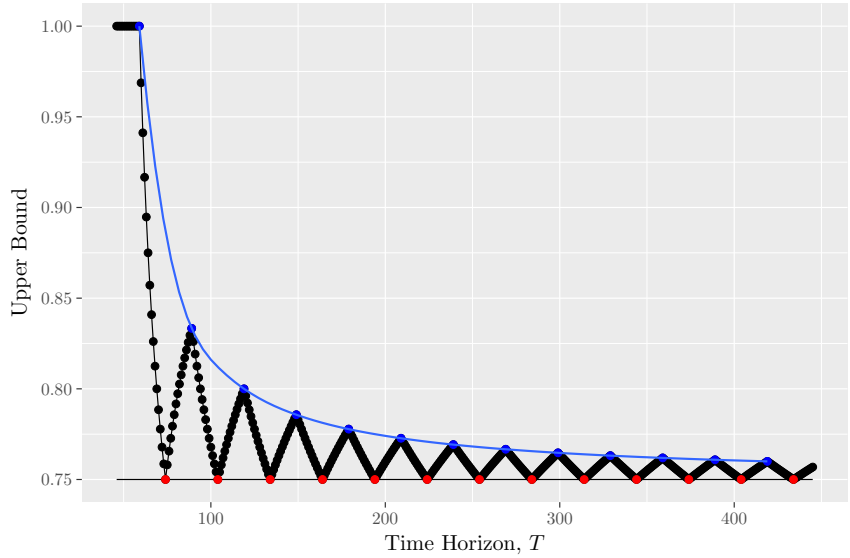
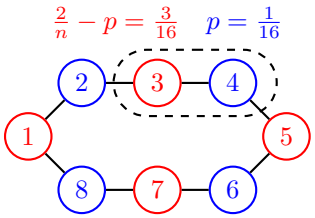


Figure 2.1: Best Upper Bound achievable under the diametric strategy

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 C.6).

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(**??**)). 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.

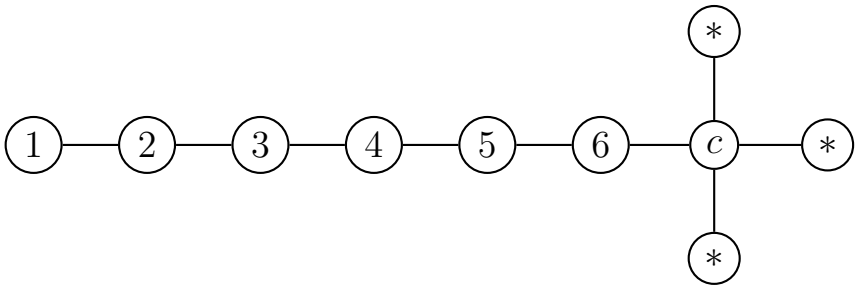


Figure 2.2: Labeling on the graph S_4^5.

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*

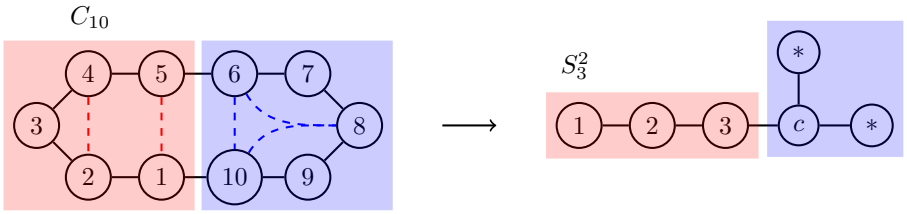


Figure 2.3: C_{10} can be simplified to S_3^2 by node identifying.

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.

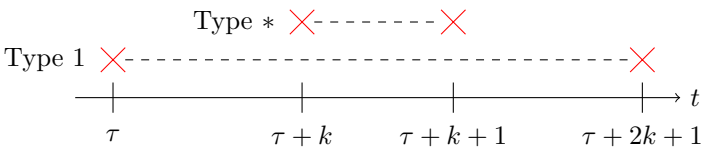
Hence we have the solution in $m \geq 2(n+k)$, so we can now restrict ourselves to $m < 2(n+k)$.

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 $2(k+1) \leq m \leq 2(n+k)$.

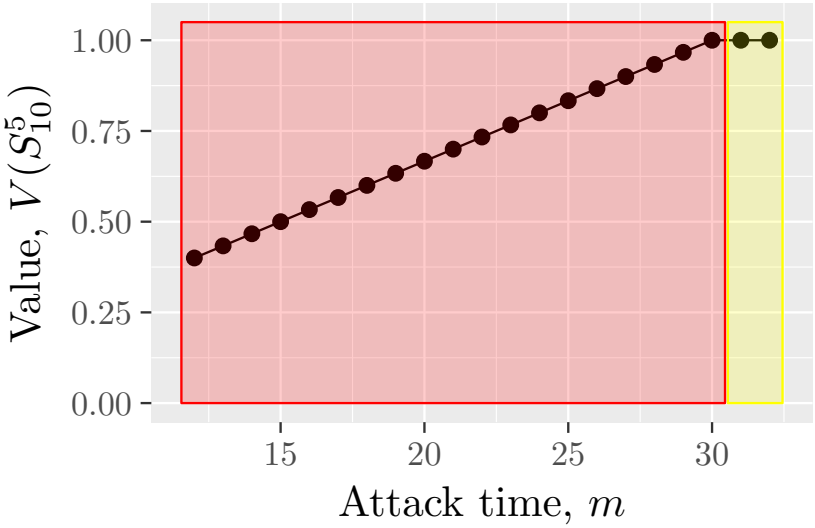


Figure 2.4: Value of the Star Graph, S_{10}^5

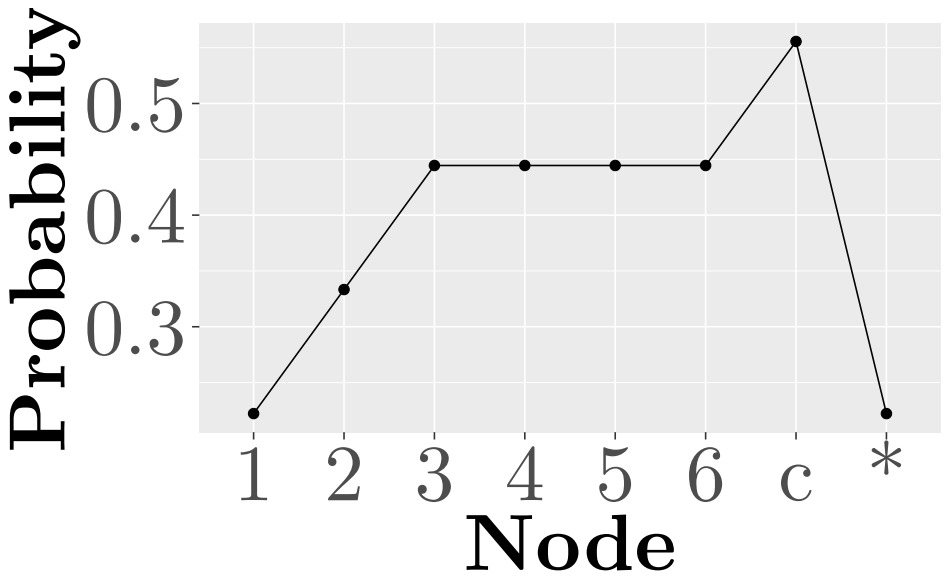
We then seek solutions in the region $m < 2(k+1)$. However in this region we are below the Random Oscillation bound for the patroller and we can suggest some improvement as in [23] from the embedded Hamiltonian bound. To see the issue and why the Random Oscillation can be improved we will look at the 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 D.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 [23] 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, $L = \left\{ i \,|\, i \leq \left\lfloor \frac{m}{2} \right\rfloor + 1, i \leq k+1 \right\}$
- Middle nodes, $M = \left\{ i \,|\, \left\lfloor \frac{m}{2} \right\rfloor + 2 \leq i \leq n+k- \left\lfloor \frac{m}{2} \right\rfloor, i \leq k+1 \right\}$
- Right nodes, $R = \left\{ i \,|\, i \geq n+k+1- \left\lfloor \frac{m}{2} \right\rfloor, i \leq k+1 \right\}$
- Star node, $S = \{c, *\}$

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

Then $V \geq w_{\min} \equiv w_{\min}^N = \min\left\{ w_{\min}^L, w_{\min}^M, w_{\min}^R, w_{\min}^S \right\}$, where $w_{\min}^X = \min_{i \in X} w(i)$. 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.

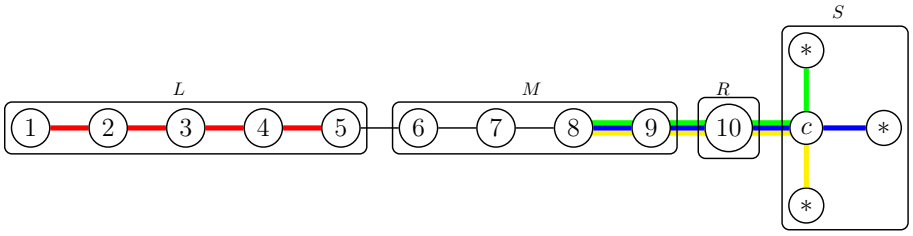
We will use P to be the probability the Random Oscillation is played and $Q = 1 - P$. We split, Q, the improvement chance into node classes, $Q = Q_L + Q_M + Q_R + Q_S$, where Q_X is the amount of probability used to improve the probability of intersection for the set X. We use q_X to be probability 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 [23]. 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 D.2).

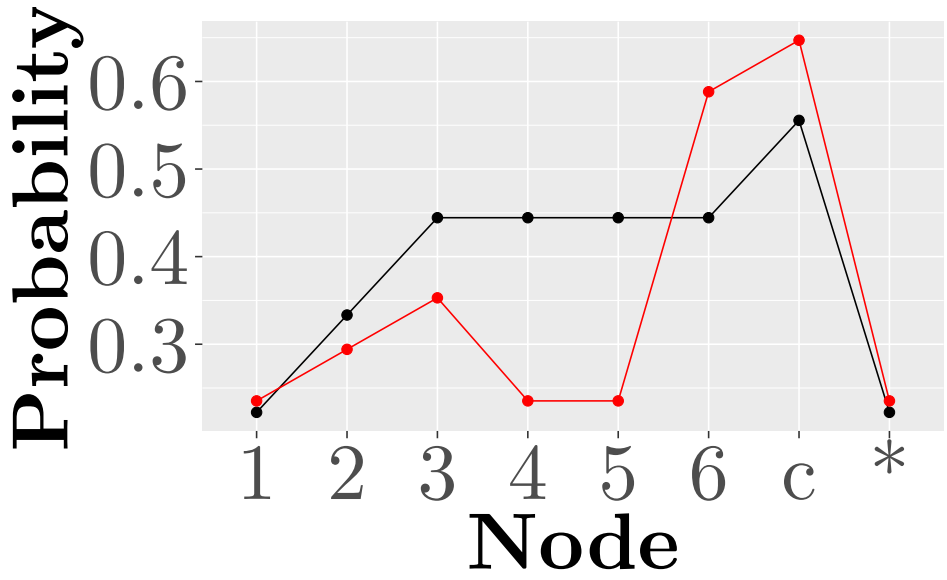


Figure 2.5: Interception probabilities of S_4^5 when $m = 4$, with the red Probabilities showing the Naive Improvement Policy $\alpha\left(\frac{2}{17}, \frac{6}{17}\right)$.

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 D.3 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 $V \geq \frac{2m}{2(n+k+m)}$

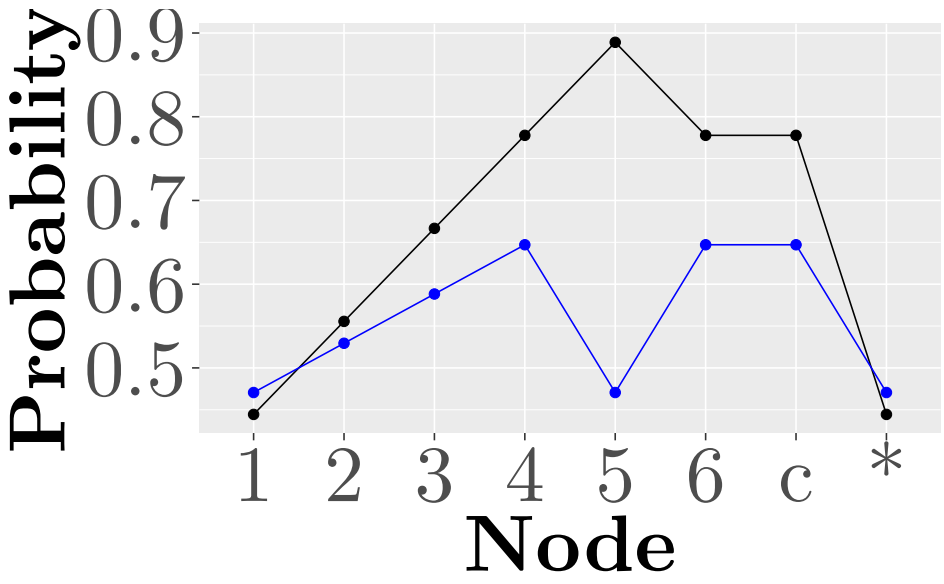


Figure 2.6: Interception probabilities of S_4^5 when $m=8$, with the blue Probabilities showing the Choosing Improvement Policy β_1 $(\frac{2}{13}, \frac{2}{13})$.

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

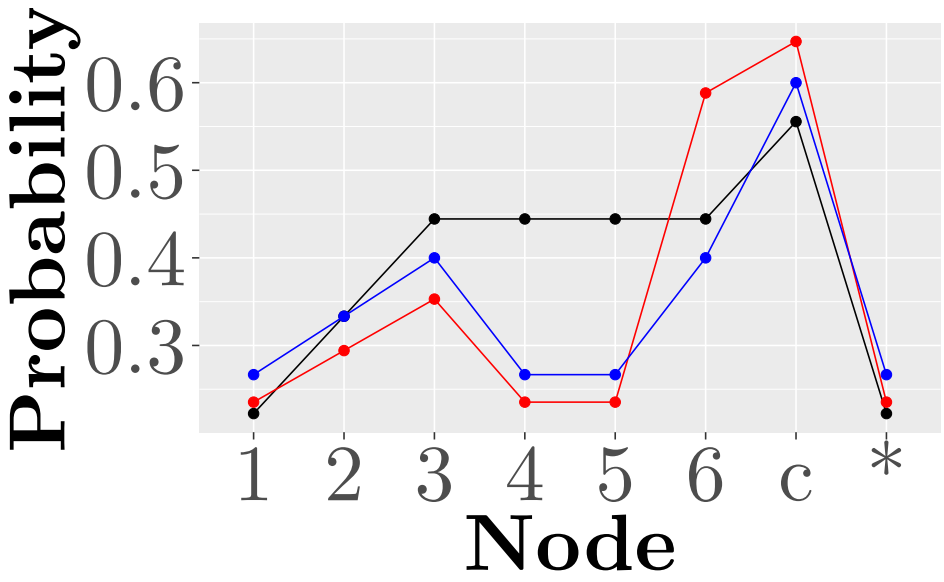


Figure 2.7: Interception probabilities of S_4^5 when $m = 4$, with the red Probabilities showing the Naive Improvement Policy α $\left(\frac{2}{17}, \frac{6}{17}\right)$ and the blue Probabilities showing the Choosing Improvement Policy β_2 $\left(\frac{1}{7}, \frac{3}{14}\right)$.

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 D.4 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 $k_{\max} \equiv \max_{i=1,2,..,h} k_i$.

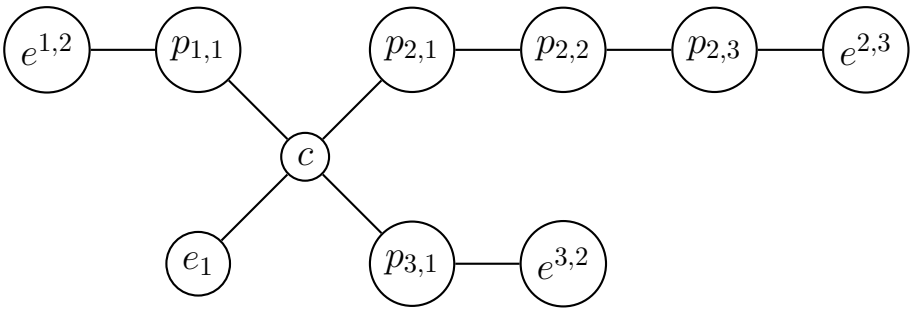


Figure 2.8: Labelling of $S_4^{1,3,1}$

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 $\tau+(k_{max}-i)+1, ..., \tau+(k_{max}+i)+1$).

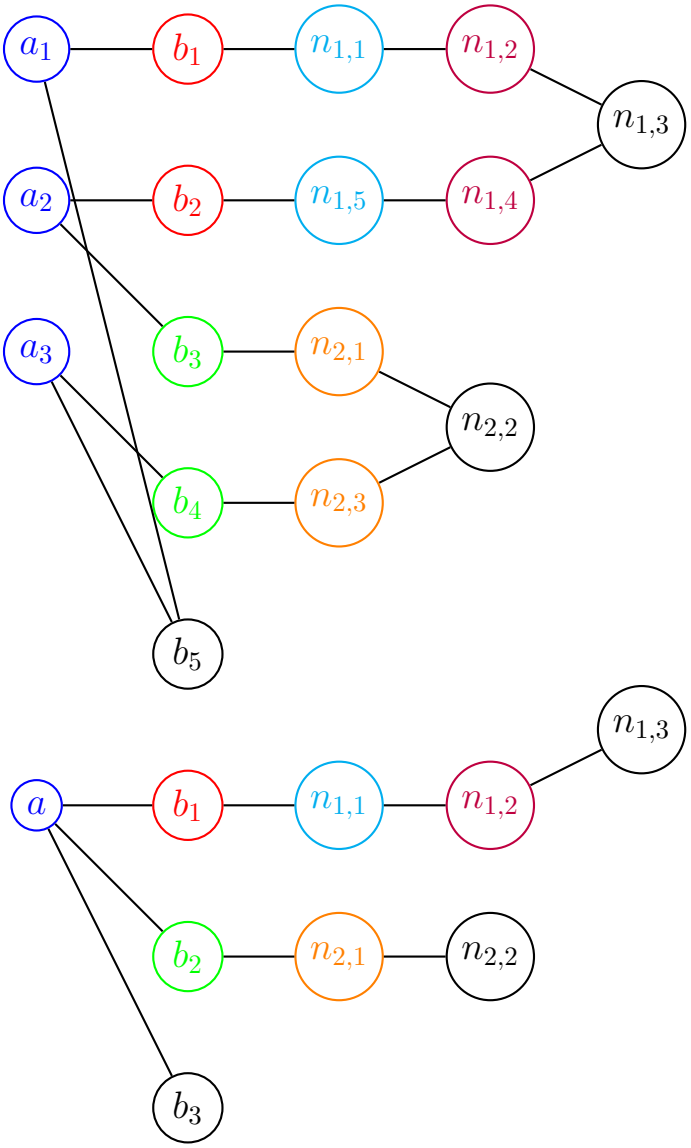
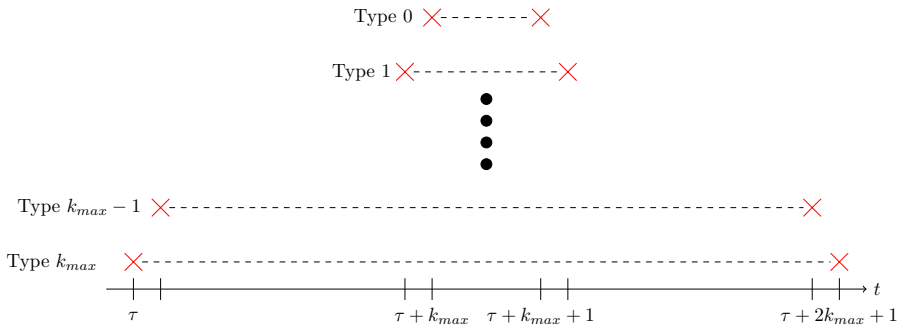


Figure 2.9: Simplification of C_{16} to $S_3^{3,2}$



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

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

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 $V = 1$.

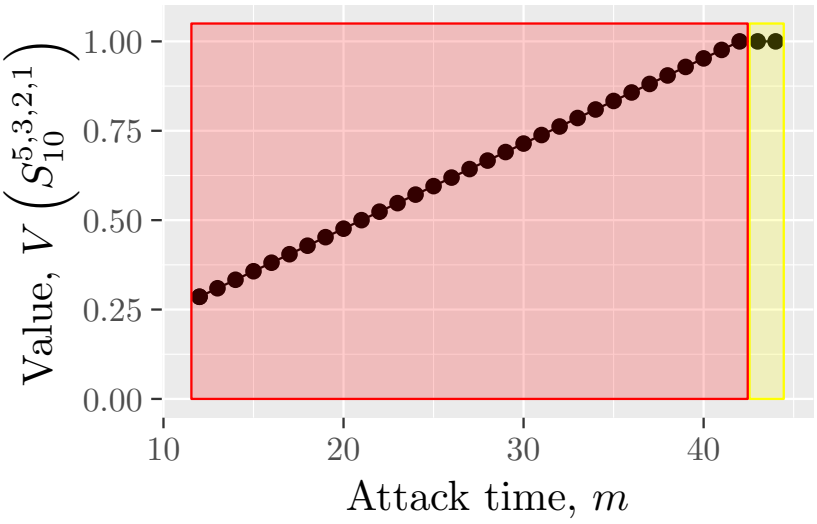


Figure 2.10: The value of the star graph, $S_{10}^{5,3,2,1}$

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)

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.

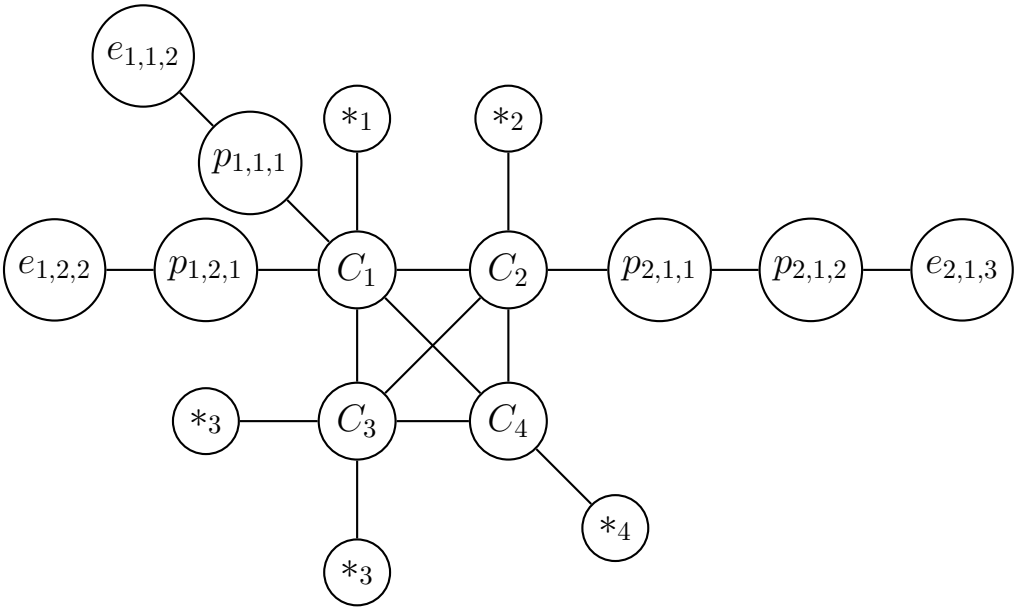


Figure 2.11: Example of labelling on $(S_3^{1,1}, S_2^2, S_2, S_1)$

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

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

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

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

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

Now under simplification of the centre's, i.e $\bigodot_{i=1}^{p} S_{n_i}^{\boldsymbol{k}_i}$ 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. □

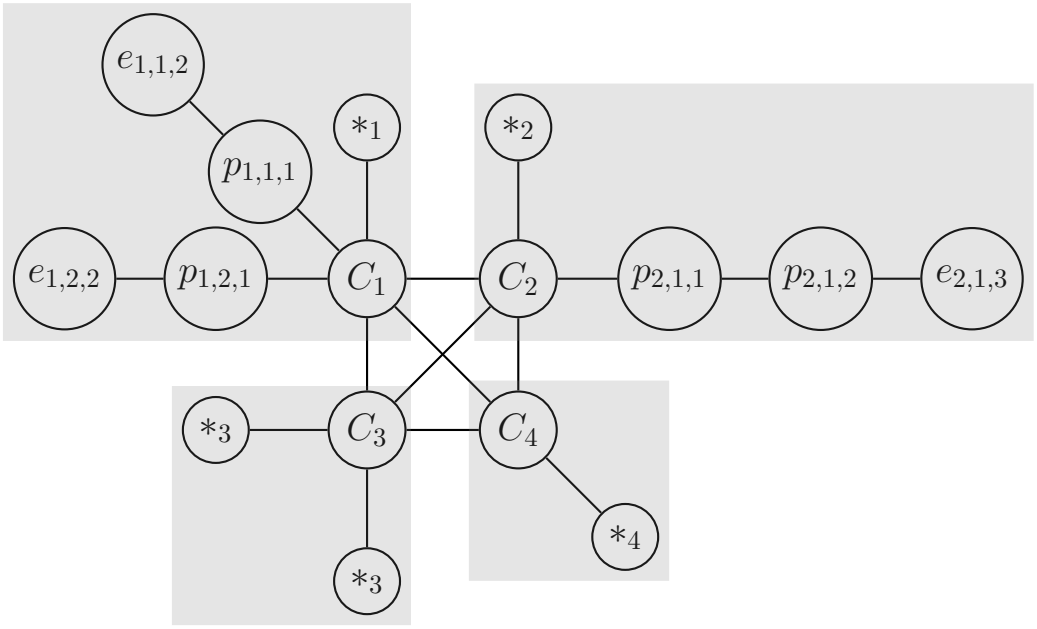


Figure 2.12: Example of decomposition on $(S_3^{1,1}, S_2^2, S_2, S_1)$

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 1.3.

3.1 Deterministic Attack time experiments

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

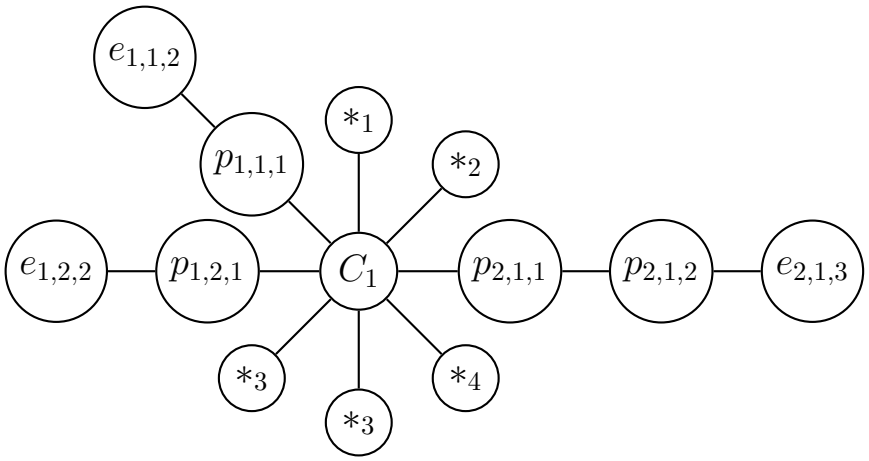


Figure 2.13: Example of simplification on $(S_3^{1,1}, S_2^2, S_2, S_1)$ to $S_8^{2,1,1}$

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

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.

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

Looking at the single node problem, we notice that there is no cost to transition to the state $s = B$, at which point we incur a cost of $c\lambda R$ to go to $s = B + 1$ and then we incur a cost of $c\lambda$.

We notice that the index has its first value in the state $s = B - 1$, so in our main problem, when deciding between nodes, we put an index on a state that isn't immediately about to incur cost. Then once its incurring costs, we place a higher index.

However consider a large such as $B = 100$, then for states $s = 1, ..., 98$ we have no index and have no incentive to go visit at all, whether this spike in the index causes any issue with the proposed heuristics is worth some consideration.

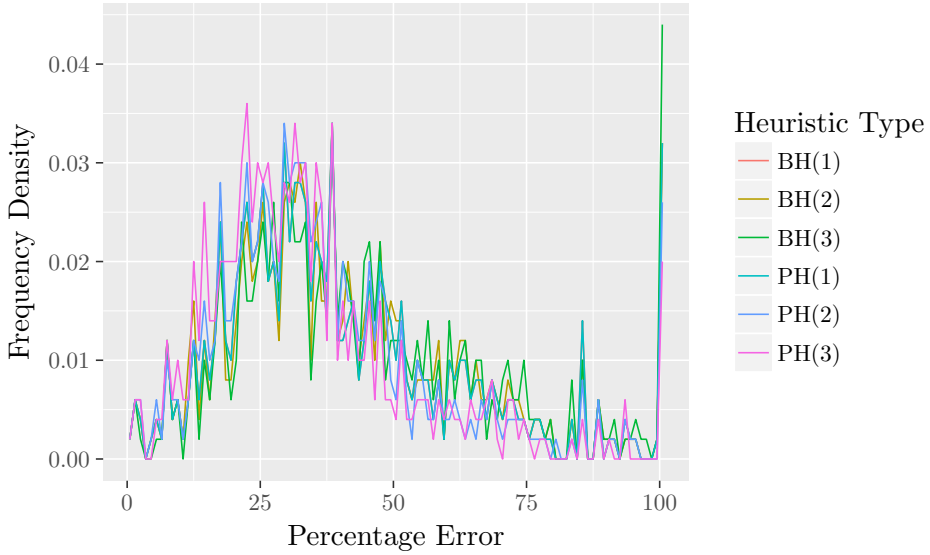


Figure 3.1: Frequency of percentage error with only deterministic attack times

3.2 Smoothing the index

To smooth the spike in 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 $s \leq B-1$, 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 $s \leq B-1$, we do not need that sum is the correct index.

Consider a the splitting function, for node j's index, $l_j(x)$ with $x \in [1, ..., B_j - 1]$ s.t $L_j(B_j-1) = B_j - 1$ 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.

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

and for the benefit use the cumulative version

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

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

We now present, at depth 3, both the equal and unequal splitting against its original counterpart for the penalty (3.2) and benefit heuristics (3.3).

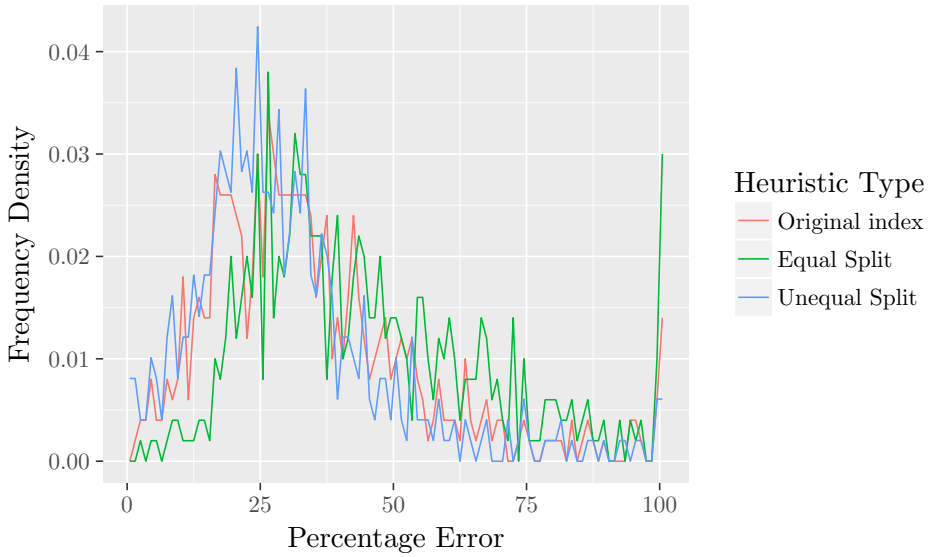


Figure 3.2: Equal and unequal splitting against original for penalty heuristic

Our numerical experiments would seem to indicate that splitting the index amongst all states, $s \leq B$ does not seem to provide any increase in the heuristics performance.

4 Strategic Patroller with Random Attackers and Local-observations

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

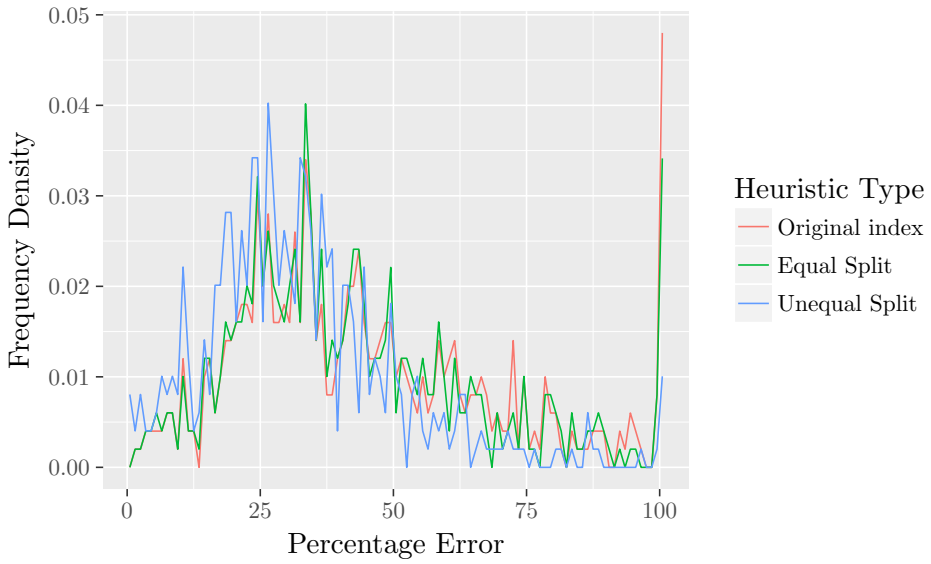


Figure 3.3: Equal and unequal splitting against original for benefit heuristic

4.1 Altering the problem for instantaneous moving patroller

We may now wish to alter the standard set-up of a patroller moving at unit speed along the edges, if the guard was flicking through live-video feeds, while unit time is spend on each feed, there is no transition time between nodes. To more closely represent this we will alter the theory to account for an instantaneously moving patroller.

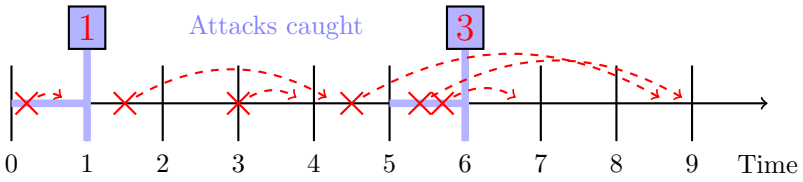


Figure 4.1: Example of timing

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

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

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

Using this new definition of $W(k)$, Theorem 1.2 still holds and gives us the optimal policy. Hence suggesting an index of

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

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.

4.3 Altering problem to accommodate local-observable information

Now that we have established a instantaneously moving patroller, we can 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, instead of waiting till the patroller leaves the node.

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

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

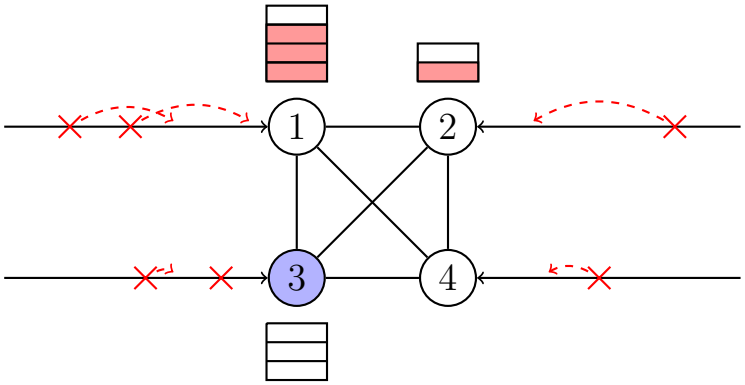


Figure 1: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

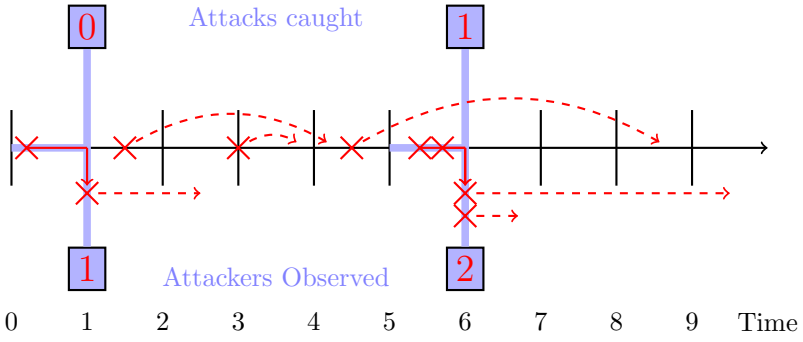


Figure 4.2: Example of timing

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

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

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

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

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

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.

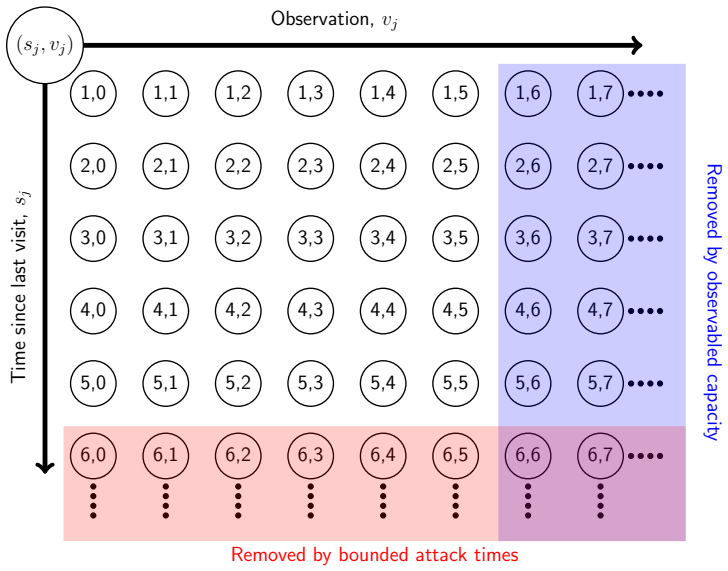


Figure 2: State space diagram for node j, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

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 F.

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 $s = B$.

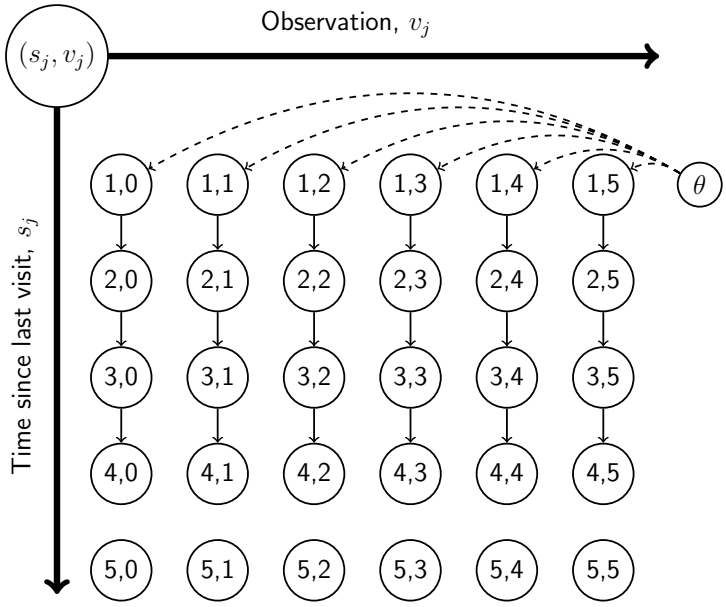


Figure 4.3: Optimal movements for $s < B$

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 $s = B + 1$.

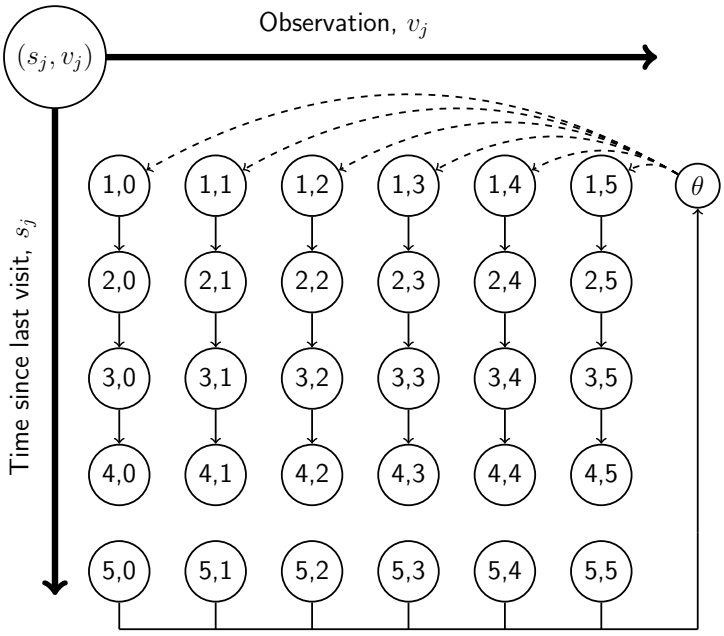


Figure 4.4: Optimal movements for $s < B$ and $s = B + 1$

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

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

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

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

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

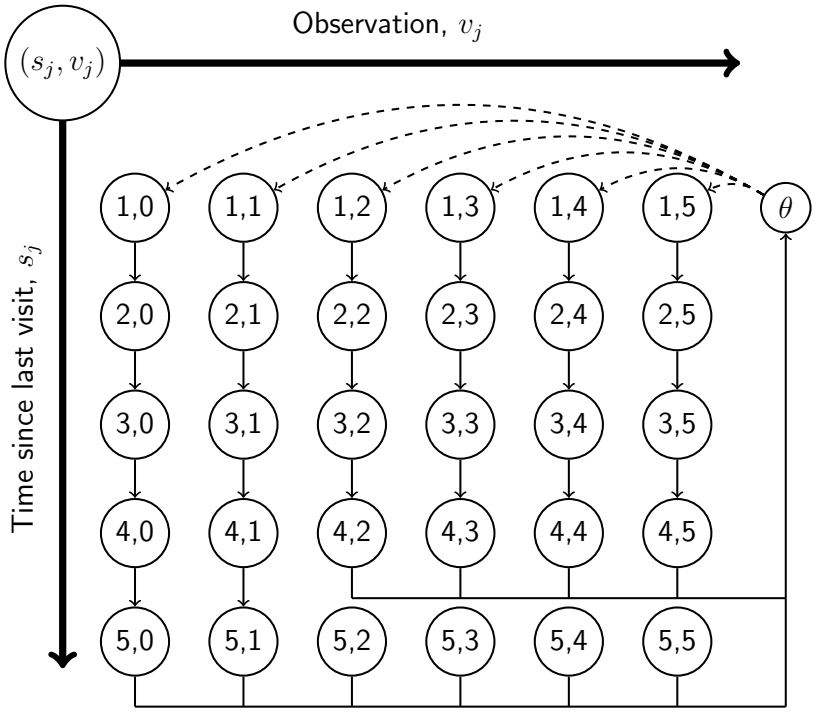


Figure 4.5: Threshold policy, $\pi_{\text{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

- $c\lambda R + cv_{\max} < g \leq c\lambda$ 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

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

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

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

Where

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

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

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

4.6 Index and heuristics

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

4.7 Numerical experiments

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

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

From Figure 4.6 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 4.7 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 [18].

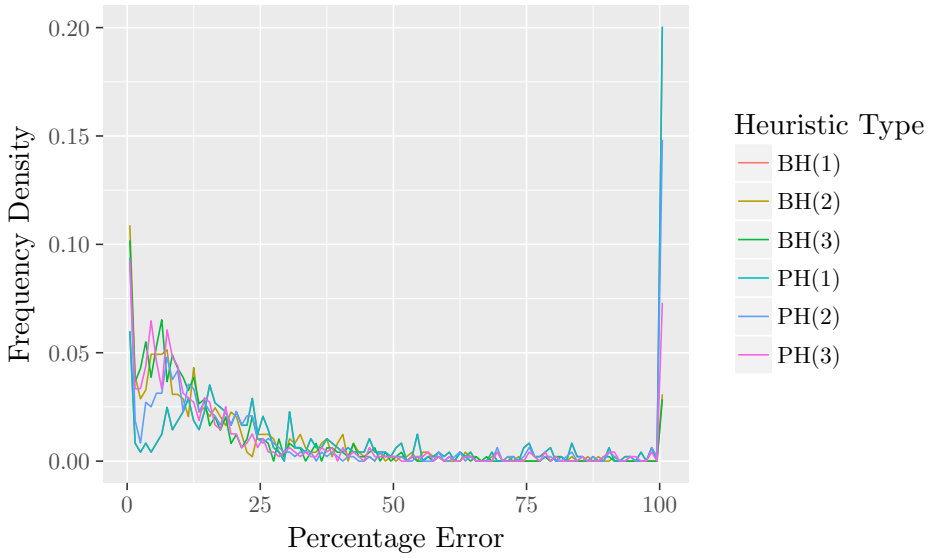


Figure 4.6: Frequency density of percentage errors made by heuristics in simulations

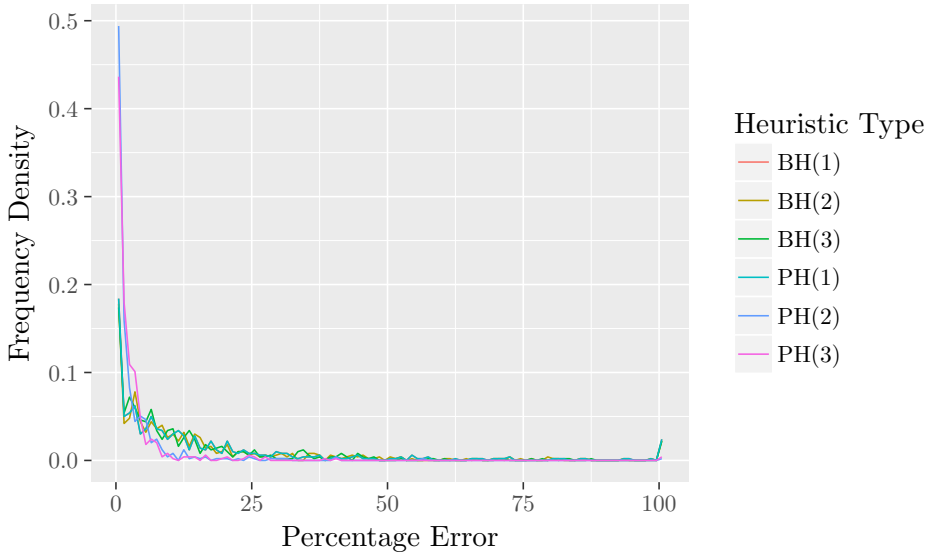


Figure 4.7: Frequency density of percentage errors made by heuristics in simulations for new index

5 Future Work

5.1 Use of trees in general graphs for patrolling games

For now we will assume that the value of the game for all tree graphs has been found. We know consider the role of spanning tree graph, $T = (N_T, E_T)$, in a general graph, $G = (N, E)$. Then we know $V(G) \leq V(T)$, as the introduction of more edges cannot harm the patroller.

As a graph may have more than one spanning tree, we may wish to find the best spanning tree, $T_{\min}(m) = \text{argmin}_T\{V(T, m) \,|\, T \text{ is a spanning tree of } G\}$.

We propose that we may take all spanning trees, $\mathcal{T}$ and there values and play the strategies suggested by these with certain probabilities, prompted by the values of the trees.

5.2 Proof completions

The proof for the time-limited diametric attack (See Appendix C.3) needs cleaning up. The idea of this proof needs extending to the time-delayed diametric attack and type-delayed diametric attack

5.2.1 Plan

Time Estimate: 1 Month

Plan: !Not sure what to go here!

5.3 Complete Numerical experiment for simultaneous moving patroller

We first focus on the deterministic case, so that

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

and an index

$$W_i(\boldsymbol{s}) \equiv \begin{cases} 0 \text{ if } s_i < B_i \\ B_i c_i \lambda_i R_i \text{ if } s_i = B_i \\ (1 + X_i) c_i \lambda_i \text{if} s_i = B_i + 1 \end{cases}$$

Time Estimate: 1 Month

Plan: To rewrite code to deal with generic attack distributions, by using functions and integration tables. To alter code to use a path selection which works out the optimal long-run average cost for the heuristic policy at a certain look-ahead, by the use of cycles. Then to run numerical experiments, with the same set up as in [18] and compare the results to the optimum. We also want to check if the same issue is found when we use purely deterministic attack times an whether splitting the index seems to help (as it does with local observations)

5.4 Extending Discrete Attack time to generic distributions for local observations

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

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

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

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

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

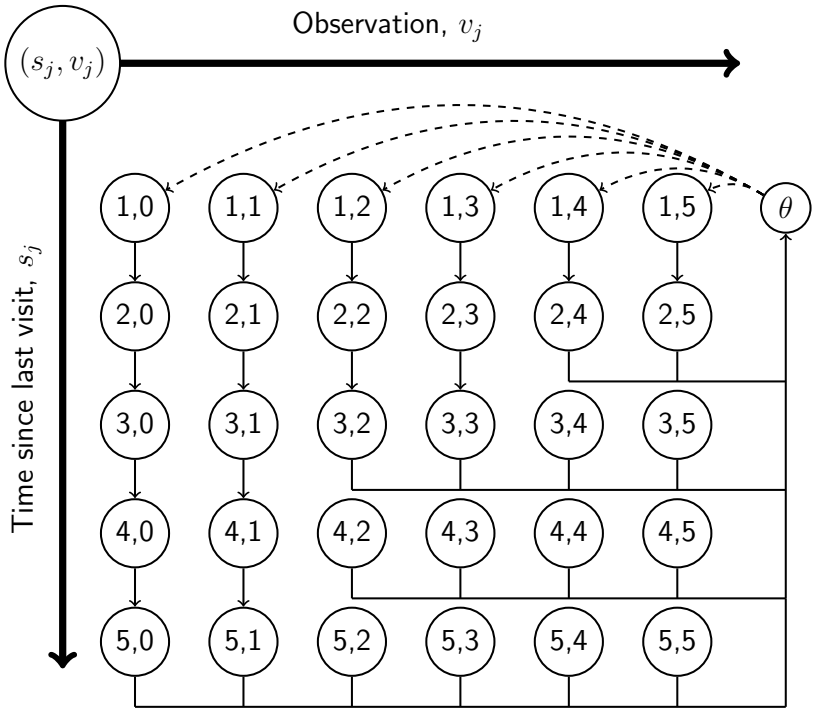


Figure 5.1: Threshold policy, $\pi_{\text{Th}}(6, 4, 2, 2, 0)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

$$\lim_{n\to\infty} V_n^{\pi_{NR}}(B,v) - V_n^{\pi_R}(B,v) \geq 0$$

$$\iff \lim_{n\to\infty} c(\lambda \int_{B-1}^{B} P(X \leq t)dt + vP(B-1 < X \leq B)) + E[V_{n-1}^{\sigma}(B+1,0)] - (\omega + E[V_{n-1}^{\sigma}(\theta)]) \geq 0$$

$$\iff g \leq c(\lambda \int_{B-1}^{B} P(X \leq t)dt + vP(B-1 < X \leq B)) = c(\lambda R_B + vp_B)$$

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)

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 $s = B+1$.

5.4.1 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.

5.5 Strategic Patroller with Random Attackers on Edges

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

We use an directed graph, $Q = (N, E)$, with nodes $N = \{1, ..., n\}$ and edges $E = \subset E_{\text{Comp}} = \{(i,j) \,|(i,j) \in N\}$. We have our adjacency matrix, A, where $A_{i,j}$ means that a transition from node i to node j is possible.

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

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

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

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

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

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

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

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 $\widetilde{s}_{i,j} = \min\{s_{i,j}+1, B_{i,j}+1\} \; \forall (i,j) \in E \backslash \{(l(\boldsymbol{s}), j')\}$.

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 1.3 and in-fact we can see a link between the edges here and the nodes in the original proble. To seek some correspondence between the two problems by using the edge-vertex dual for a directed graphs (as in).

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

Do I need to show this 1-1 correspondence ?

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

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

5.5.1 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.

5.6 Investigate weighted graphs

We may wish to incorporate the idea of distance between nodes, which is not of unit length as proposed in the earlier formulations. We would also like to introduce a concept of the patrollers speed into the model as well.

5.6.1 Plan

Time Estimate: 2 Months

Plan:

5.7 Investigate types of patrolling

Further to the idea that the patroller may overlook an attacker ([19]), we may propose that not only can the patroller make the decision of where to move to, but also in what 'mode' they search the region, fast or slow, each with different chances of overlook.

5.7.1 Plan

Time Estimate: 3 Months

Plan:

References

[1] Mohammad J. Abdel-Rahman and Marwan Krunz. Game-theoretic quorum-based frequency hopping for anti-jamming rendezvous in DSA networks. *2014 IEEE International Symposium on Dynamic Spectrum Access Networks, DYSPAN 2014*, pages 248–258, 2014.

[2] Steve Alpern, V. J. Baston, and Skander Essegaier. Rendezvous search on a graph. *Journal of Applied Probability*, 36(01):223–231, mar 1999.

[3] Steve Alpern and Robbert Fokkink. Accumulation games on graphs. *Networks*, 64(1):40–47, 2014.

[4] Steve Alpern, Robbert Fokkink, Marco Timmer, and Jerome Casas. Ambush frequency should increase over time during optimal predator search for prey. *Journal of the Royal Society Interface*, 8(64):1665–1672, 2011.

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

[6] 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.

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

[8] Steve Alpern and Lim Wei Shi. The Symmetric Rendezvous-Evasion Game. *SIAM Journal on Control and Optimization*, 36(3):948–959, 1998.

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

[10] William L. Black. Discrete sequential search. *Information and Control*, 8(2):159–162, 1965.

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

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

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

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

[15] 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.

[16] Kensaku Kikuta and William H. Ruckle. Continuous accumulation games on discrete locations. *Naval Research Logistics*, 49(1):60–77, 2002.

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

[18] 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.

[19] 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.

[20] Kyle Y Elsevier Lin, Kyle Y Lin, and Dashi I Singham. Calhoun: The NPS Institutional Archive DSpace Repository Finding a hider by an unknown deadline Finding a hider by an unknown deadline. *Operations Research Letters Calhoun Operations Research Letters*, 441(44):25–32, 2016.

[21] 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.

[22] David Matula. A Periodic Optimal Search. *American Mathematical Monthly*, 71(1):15–21, 1964.

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

[24] 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.

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

[26] W. H. Ruckle and K. Kikuta. Continuous accumulation games in continuous regions. *Journal of Optimization Theory and Applications*, 106(3):581–601, 2000.

[27] William H Ruckle. *Geometric games and their applications*. Research notes in mathematics. Pitman Advanced Pub., 1983.

[28] 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.

[29] N Zoroa and P Zoroa. A Game Related to the Number of Hides Game. *Journal of Optimization Theory and Applications*, 103(2):457–473, 1999.

[30] P Zoroa and N Zoroa. Search and Ambush Games with Capacities. *Journal of Optimization Theory and Applications*, 123(2):431–450, 2004.

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.

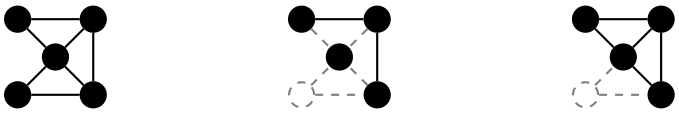
Note. For notational purposes both $e_{i,i'} \in E$ and $(i, i') \in E$ are equivalent.

Definition A.2 (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.3 (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.4 (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.5 (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 $(n_1, n_2) \in E \implies (n_1, n_2) \in E'$

Definition A.6 (Edge types)**.** An edge (i, i) is called a *loop*. If more than one copy of the edge (i, j) exits in E, it is said to be a *multiple edge*. A *simple graph* is graph without any loops or multiple edges.

Definition A.7 (Node Identification)**.** The operation of *node identification* on two nodes, u and v, of a graph, $G = (N, E)$ into a single node w, is a mapping $f : N \to N'$ resulting in a new graph $G' = (N', E')$ where $N' = (N \backslash \{u, v\}) \cup \{w\}$ with $E' = E \setminus \{(u, v)\}$ if $(u, v) \in E$ and under the condition that $\forall x \in N$, $f(x) \in N'$ is incident to $e' \in E'$ iff $e \in E$ is incident to $x \in N$. Furthermore if a graph, Q, undergoes repeated node identification to become Q' then we say it has been *simplified*.

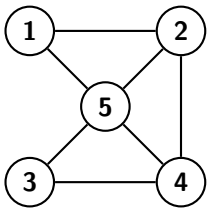


Figure 3: Graph, Q

Definition A.8 (Embedding)**.** An *embedding* of a graph, G, onto a surface(compact, connected 2-manifold), Σ, is a representation of G in Σ in which points are associated with nodes and simple arcs(homeomorphic images of $[0,1]$) are associated with edges such that

- the end points of arc are the points associated with the nodes indecent with the edge
- no arcs include points associated with other nodes
- two arcs never intersect at a point which is interior to either of the arcs

Definition A.9 (Planar)**.** A graph, G, is called *planar* if it can be embedded in the plan $\mathbb{R}^2$.

Definition A.10 (Distance,diameter)**.** The *distance* between node i and nodei' is the minimum number of edges between i and i' denoted, $d(i,i')$. A graph's *diameter* is the maximum distance between any pair of nodes, defined by $\bar{d} \equiv \max_{i,i' \in N} d(i,i')$. Any pair of nodes i,i' that are the diameter apart(i.e $d(i,i') = \bar{d}$) are called *diametrical*

Definition A.11 (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.12 (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.13. For the graph Q as in Figure 3:

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

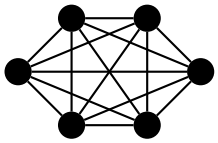
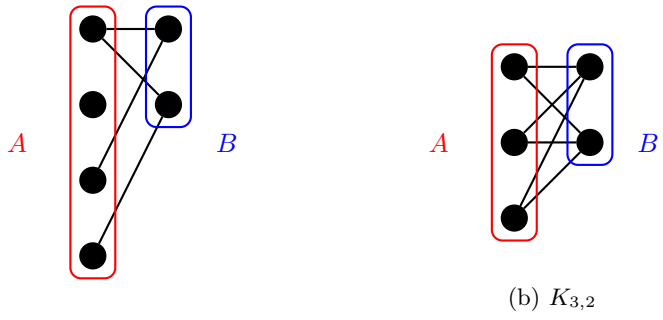


Figure A.2: The complete graph of 6 nodes,K_6.

Definition A.15 (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.16 (Complete bipartite)**.** The *complete biparite graph*, $K_{a,b}$, is a bipartite graph of $a + b$ nodes (where $|A| = a$,$|B| = b$), in which all edges are present, i.e $e_{i,i'} \in E \; \forall i \in A \, \forall i' \in B$ and $e_{i,i'} \in E \; \forall i \in B \, \forall i' \in A$.



(a) Q

(b) $K_{3,2}$

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

Definition A.17 (Subdivision,Smoothing)**.** A *Subdivision* (or *expansion*) of a graph, G, is a new graph G' which is made by subdividing a chosen edge. The subdivision of an edge $\{u, v\}$ yields a graph with a new node w and the splitting of the edge $\{u, v\}$ into $\{u, w\}$ and $\{w, v\}$.

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

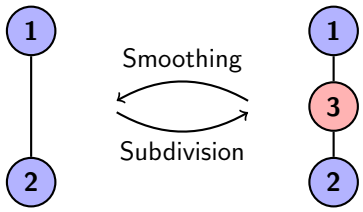


Figure A.4: Subdivision and Smoothing of the edge $\{1, 2\}$

Definition A.18 (Edge-vertex dual)**.** A *edge-vertex dual* of a directed graph G, called $EV(G)$, is made of a vertex set $V_{EV(G)} = E_G$ and whose edge set is made up of a directed edge between $e_1, e_2 \in V_{EV(G)}$ if in G the edge e_1's head meets the tail of e_2 at some node.

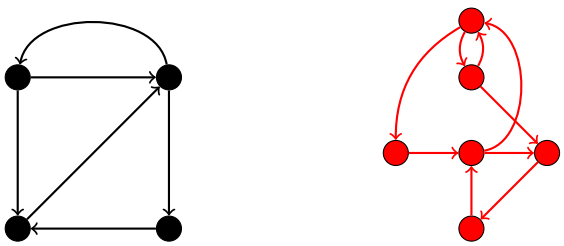


Figure A.5: A graph G and its directed edge-vertex dual, $EV(G)$

B Examples

B.1 Example of decomposition

Example B.1. For Q as seen in Figure B.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 [7]).

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

B.2 Example of simplification

Example B.2. For Q as seen in Figure B.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 $V(Q) \leq \frac{3}{4}$

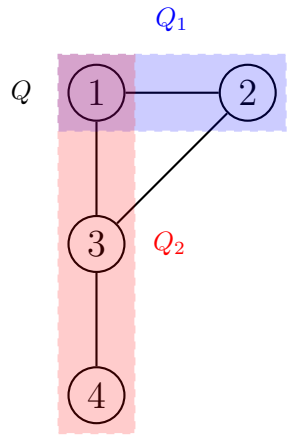


Figure B.1: Decomposition of Q into Q_1 and Q_2.

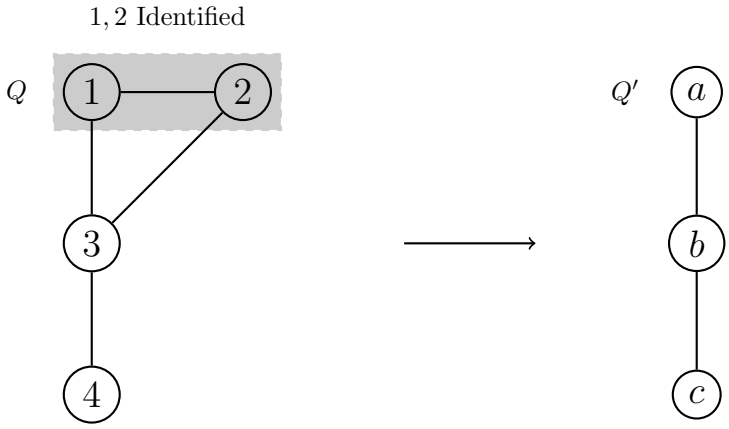


Figure B.2: Simplifcation of Q to Q' by identification.

C Patrolling games

C.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, $C_0 = \min(t, T-m+1)$, $C_i = \left(\min(t+i\bar{d}, m, T-(t+i\bar{d})\right)_+$ 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.

C.2 Proof of conditions on T for diametric attack

Proof. Using $T = m-1+(k+1)\bar{d}$ in the formula gives, $\alpha = \left\lfloor \frac{(k+1)\bar{d}-m}{\bar{d}} \right\rfloor = (k+1) + \left\lfloor \frac{-m}{\bar{d}} \right\rfloor = (k+1)-2 = (k-1)$ (the final part is because $2 > \frac{-m}{\bar{d}} \geq -1$ and we will assume that $m > \bar{d}$ here otherwise waiting at one node is just as good as the bound we are trying to achieve)

$m-\bar{d}+(m+m(k-1))_+ + \big(m-1-(m-1+(k-1+1)\bar{d})\big)_+ + \big(m-1-(m-1+(k+1)\bar{d})\big)_+$

which is $m-\bar{d}+(mk)_+ + \big((k+1)\bar{d}-k\bar{d}\big)_+ + \big((k+1)\bar{d}-(k+1)\bar{d}\big)_+$ giving $m-\bar{d}+mk+\big(\bar{d}\big)_+ + (0)_+ = m(k+1)$. Giving the fraction of $\frac{m(k+1)}{2(k+1)\bar{d}} = \frac{m}{2\bar{d}}$.

For the second part, first we seek to prove that within the choice of T from $m-1+(k+1)\bar{d}+r$ where $0 \leq r < \bar{d}$ is the maximum when $r = m$ (i.e $T = 2m-1+(k+1)\bar{d}$).

As the choice of r only affects the final 3 parts (middle and ends values), we can just look at considering these values and seeing what the maximal choice is.

Upon substitution we get that: $\alpha = \left\lfloor \frac{(k+1)\bar{d}+r-m}{\bar{d}} \right\rfloor = (k+1) + \left\lfloor \frac{r-m}{\bar{d}} \right\rfloor$

so formula becomes $(m(\alpha+1))_+ + \big((k+1)\bar{d}+r-(\alpha+1)\bar{d}\big)_+ + \big((k+1)\bar{d}+r-(\alpha+2)\bar{d}\big)_+$. To decide r we need to know if middle values or end values are non-zero.

Note. The second end value will never be non-zero as $(k+1)\bar{d}+r-(\alpha+2)\bar{d} = (k+1)\bar{d}+r-((k+1)+\left\lfloor \frac{r-m}{\bar{d}} \right\rfloor+2)\bar{d} = r-(\left\lfloor \frac{r-m}{\bar{d}} \right\rfloor+2)\,\bar{d} < r-(-1+2)\bar{d} = r-\bar{d} < 0$.

- No middle values and no end values is impossible assuming $k \in \mathbb{N}_0$.
- Middle values but no end values. As we really want to maximize the end value, increasing r up to the point where α increases (giving a raise of m attacks captured) but increases the number of total attacks by 2 each time it is raised. Hence minimal r is chosen to increase α. This is when $r-m = -\bar{d}$ i.e $r = m-\bar{d}$ changes α by 1 (as $r < m-\bar{d}$ gives an -1 to α, but critical point is when equal to).
- Middle values and end value. As we are looking at $(m-\bar{d})(\alpha+1) + (k+1)\bar{d}+r$, we still want to increase α without increasing the number of attacks too much, i.e as above.

Then we show that the maximal subsequence tends to the bound as $T \to \infty$, i.e as $k \to \infty$. When substituted, we get that $\alpha = (k+1)$ and so the formula becomes $m-\bar{d}+(m+m(k+2))_+ + \big(2m-1-(m-1+(k+2)\bar{d})\big)_+ + \big(2m-1-(m-1+(k+3)\bar{d})\big)_+$ giving $m-\bar{d}+(k+3)m+\big(m-(k+2)\bar{d}\big)_+ + \big(m-(k+3)\bar{d}\big)_+$. As $m < 2\bar{d}$ then we get $m(k+4)-\bar{d}$ caught out of $2(m+k\bar{d})$ giving a fraction of $\frac{m(k+4)-\bar{d}}{2(m+k\bar{d}} \to \frac{m}{2\bar{d}}$.

Hence as the maximal subsequence tends down to the bound, it implies the result. □

C.3 Proof of time-limited diametric attack

Proof. First consider all the pure patrolling strategies, $W_i \in \mathcal{W}$, Then as the attacker is only attacking two ends, henceforth called n_1 and $n_{\bar{d}}$, any patrol not

starting at n_1 or $n_{\bar{d}}$ is dominated by one that does. This is because the patrol will not capture any attacks until they visit either n_1 or $n_{\bar{d}}$, and then capture a set of attacks that started there previously. The patrol might as well wait there up until this point and do at least as good as arriving there for the first time.

Formally, assume that n_1 is the end node first reached by a patrol, $W(t)$ at time, $t_1 = \min\{t \,|\, W(t) = n_1\}$, then we can form the patrol, $U(t) = \begin{cases} n_1 \text{ for } t \leq t_1, \\ W(t) \text{ for } t > t_1. \end{cases}$ and $P(U, \phi) \geq P(W, \phi)$ where ϕ is the timed diametric attack (or infact the normal diametric attack.

Now we are restricted to patrols starting at end points, it is similar to see when leaving an end point, there is no other decision as you must travel to the other end point, assumed to be $n_{\bar{d}}$. Hence the question becomes when to leave n_1 and travel to $n_{\bar{d}}$. Obviously it should only be undertaken if the journey can be made and more attacks can be caught by doing so.

WLOG assume that $\tau = 0$ (other just wait longer initially, as attacks haven't started), then our choice is what leaving time (last time before moving): $t_l \in \{0, 1, ..., m-2\}$, to pick to maximize the number of attacks caught; or $t_L = \infty$, never leaving to get $\bar{d}$ attacks.

Leaving: Choosing $t_L \in \{0, 1, ..., m-2\}$ gives the patroller $\frac{m}{2\bar{d}}$ as,

Leaving at t_L gives us $\min(t_L + 1, \bar{d})$ attacks caught at n_L, and $\min(m + \bar{d} - 2 - (t_L + \bar{d}) + 1, \bar{d}) = \min(m - 1 - t_L, \bar{d})$.

Now choosing $t_L > \bar{d} - 1$, doesn't improve the first value and possibly lowers the second value. Hence we restrict ourselves to leave if we catch all attacks, i.e $t_L \leq \bar{d} - 1$. Now in this region lowering t_L lowers it by 1 and raises it only raises the second on by 1 if $m - 1 - t_L \leq \bar{d}$ (i.e $t_L \geq m - 1 - \bar{d}$ or any t_L if $m - 1 - \bar{d} \leq 0$). Hence any choice of $\left(m - 1 - \bar{d}\right)_+ \leq t_L \leq \bar{d} - 1$ is equally as good. This gives a number of attacks caught as $t_L + 1 + m - 1 - t_L = m$ out of $2\bar{d}$ placed attacks. Hence giving $V \leq \frac{m}{2\bar{d}}$.

Staying: Choosing $t_L = \infty$ gives the patroller $\frac{1}{2}$

Hence as the patroller can pick from these two options, it gives $V \leq \max\{\frac{1}{2}, \frac{m}{2\bar{d}}\}$. More explicity it gives $V \leq \frac{1}{2}$ if $m < \bar{d}$, and $V \leq \frac{m}{2\bar{d}}$ is $m \geq \bar{d}$. □

C.4 Proof of time-delayed attack

Proof. First Consider a patrolling strategy that is at a $*$ node at time $t \geq k$ (i.e $*$ node attacks have begun) and seek to show that staying amongst the $*$ nodes until the attacks end is at least as good as moving to node 1 and waiting, and possibly returning to $*$ (if time allows).

We will consider two cases of $m \leq 2(k+1)$ and $m > 2(k+1)$.

1. In this case we will first show that returning to $*$ type nodes is never an option once it is left, consider leaving at t, then the first possible return would be $t + 2(k + 2) \geq k + 2(k + 2) > k + m$ and hence all the attacks would be caught and there would be no point returning. Hence the only option in this scenario is whether it is worth it leave at this point t and go to node 1 and wait until the end of the game.

   If she was to stay around the end nodes then from this point onwards (not including the node we are at , at time t) we would get a payoff of $\frac{k+m-t-1}{2(n+k)}$.

   This is becuase we either get:

$$k + m \text{ odd } \underbrace{\frac{1}{n+k} + ... + \frac{1}{n+k}}_{\frac{k+m-t}{2}\text{times}}$$

$$k + m \text{ even } \underbrace{\frac{1}{n+k} + ... + \frac{1}{n+k}}_{\frac{k+m-1-t}{2}-1\text{times}} + \frac{1}{2(n+k)}$$

   In either case the payoff for moving about at the $*$ type nodes is as above.

   Now consider moving away to 1, which will be reached at time $t + k + 2$ and as we must wait here the payoff depends on a few things. It is

$$\frac{\min(m, t + k + 2, m - (t + k + 2 - (2k + 1))}{2(k + 1)} \times \frac{k + 1}{n + k} = \frac{\min(m, t + k + 2, k + m - t - 1)}{2(n + k)}$$

   now note as $t \geq k \implies m > k + m - t - 1$ and as $m \leq 2(k + 1) \implies t + k + 2 \geq m$. Moreover this implies we will be in the final stretch of attacks and no new attacks will be taking place, so no more attacks will be claimed by waiting here (though moving back is just as fruitless)

   Hence in this case the payoff is $\frac{k+m-t-1}{2(n+k)}$. The exact same benefit as to staying around the $*$ nodes, hence both moving and staying are equally as good, so once left a $*$ node she will have to wait at 1 (and infact the game will be over) and we have no incentive not to do it (INFACT IF THE CONDITION ABOUT LIMITED number of $*$ nodes is brought up it is infact the best option).

   Hence for any $t \geq k$ when we are at a $*$ node we might as well move to 1 and end the game.

   Now consider being at $*$ for some time $s \leq k - 1$, then we can decided to wait to time k and then make the decision as above and move to 1 or we can move immediately to 1.

   Waiting to time k gives us $\frac{1}{2(n+k)} + \frac{k+m-k-1}{2(n+k)} = \frac{m}{2(n+k)}$.

   Now leaving at time s means we arrive at 1 at time $s + k + 2$, however, if this is the plan then it is clear that starting at 1 is optimal (which we will deal with next).

   Now starting at 1 at time 0, consider the decision to move to 1 at time q (under which the decision will complete the game, as moving get us there

at time $q+k+2 \geq k$ so the decision to move back immediately is chosen), This means we get a payoff of

$$\frac{q+1}{2(n+k)}+\frac{1}{n+k}+\frac{m-(q+2(k+2)-(2k+1))}{2(n+k)} = \frac{q+1}{2(n+k)}+\frac{1}{n+k}+\frac{m-q-3}{2(n+k)} = \frac{m}{2(n+k)}$$

With the knowledge that the choice of q's choice to achieve this, if $q \geq m-2$ is chosen it is worse than the sum as, not as good on arriving and nothing gained on coming back, hence it only achieves $\frac{q+1}{2(n+k)}$, so $q = 2k+1$ might as well be chosen for $\frac{k+1}{n+k}$ (note. here that $q = 2k+1$ is always in this zone as $m \leq 2(k+1)$)

Hence it is best to wait for all time at node 1 and achieve $\frac{k+1}{n+k} \geq \frac{m}{2(n+k)}$ (for $m \leq 2(k+1)$)

2. When $m > 2(k+1)$, we shall again first consider starting at a $*$ at time $t \geq k$, however now it is not possible to state that once it is left that it can never be returned to. We care about what to do between t and $k+m$, now we will first seek to show that moving and waiting at 1 is just as good as moving around $*$ types.

   Moving purely around $*$ nodes will get us as before $\frac{k+m-t-1}{2(n+k)}$. Moving and waiting at 1 gets us $\frac{2(k+1)-(t-m+k+1)_+}{2(k+1)} \times \frac{k+1}{n+k} = \frac{k+m-t-1}{2(n+k)}$. (or $\frac{2(k+1)}{2(n+k)} = \frac{k+1}{n+k}$, if they go early enough to catch all attacks)

   The only idea that could possibly be better is to move to 1 and then wait for period of time, say q times waiting, and then move back. This would yeild

$$\frac{q+\min(2(k+1), t+k+2, t+k+2-(t-m+k+1))}{2(k+1)} \times \frac{k+1}{n+k} = \frac{q+2(k+1)}{2(n+k)}$$

   ($+\frac{1}{n+k}$ or $+\frac{1}{2(n+k)}$ if arriving before all attacks at $*$ have completed) from being there for a period and then remake the decision of what to do from $t+q+2(k+2)$. During this time period between t and $t+q+2(k+2)$ (assuming the game at $*$ is not over yet) then we will get

$$\underbrace{\frac{1}{n+k}+...+\frac{1}{n+k}}_{\frac{q+2k+2}{2}\text{times}} = \frac{q+2k+2}{2(n+k)}$$

   ($+\frac{1}{n+k}$ or $+\frac{1}{2(n+k)}$ at end)

   Hence it is better to move an wait at 1 then return if we want to, however returning serves no purpose as we will leave immediately. Hence moving to 1 and waiting is the best option if $t \geq k$. Hence consider starting at a $*$, it would be either wait till k and move back or move earlier. Moving earlier would mean that she might as well have started at node 1.

   Hence the only option for starting at node $*$ is to get $\frac{1}{2(n+k)} + \frac{k+m-t-1}{2(n+k)}$ (or $\frac{1}{2(n+k)} + \frac{k+1}{n+k}$ if all attacks are caught at node 1 at time $2k+2$ (i.e $m > 2k+3$)).

So starting at $*$ means getting $\frac{k+1}{n+k} + \frac{1}{2(n+k)}$

Let $m = 2(k+1) + r$ for $r \in \mathbb{N}$. Then considering starting at $*$ nodes, then we must decide to move to 1 or wait till k then make a decision. However deciding to move to 1 , means we might as well have started at 1.

If we at a $*$ node at some time $t \geq k$ (let $t = k + t_e$), then we can decide to (wait only if $t = k$ until $t = k + 1$) move to another $*$ node arriving at $t + 2$ or move to 1 arriving at $t + k + 2$. Now we aim to show that the option of moving to another $*$ node is not strictly better. Assume it is then it is a repeated action until time $k + m - 2 = 3k + r$ or $k + m - 1 = 3k + r + 1$ (depending on which parity we are in).

We will get a payoff of $\frac{3k+r-t}{2(n+k)} = \frac{k+m-2-t}{2(n+k)}$ or $\frac{3k+r+1-t}{2(n+k)} = \frac{k+m-1-t}{2(n+k)}$ (depending on parity). Then it will have to move to node 1 arriving at $2k + 2 + m$ (and the game will be over)

Now consider moving to 1 and waiting , arrive at time $t+k+2 = 2k+2+t_e$. Then the payoff is $\frac{\min(2(k+1),k+m-t-1)}{2(n+k)}$. Which of these is chosen as the minimum will be decided by whether they still arrive in time to collect the first few attacks (i.e it depends on the second part $k + m - t - 1 = 2k + r - t_e + 1$ which means if its greater than $2k + 2$ i.e depending on the distant between $r - 1 - t_e$)

More explicitly the term $k + m - t - 1 \geq 2k + 2$ if $r - 1 - t_e \geq 0$, in this case however it is possible to make up the difference of $r-1-t-e$ by only waiting till $2k + 1$ (we will be there at $t + k + 2 \geq 2(k + 1)$, so will leave immediately back) then returning to the $*$ at $3k + 4$ this is in comparison to $k + m = 3k + 2 + r$ meaning that we get an additional payoff depending on r's parity. If r is odd then we gain $\frac{1}{2(n+k)} + \frac{r-2}{2(n+k)} = \frac{r-1}{2(n+k)}$. If r is even then we gain $\frac{r-1}{2(n+k)}$.

So we will be on $\frac{2(k+1)+r-1}{2(n+k)} = \frac{m-1}{2(n+k)}$ which is better than $\frac{k+m-1-t}{2(n+k)}$ and hence moving to 1 and moving back at $2k + 2$ (arriving back at $3k + 4$) is better.

Similarly in the case of $2k + 2 > k + m - t - 1$ if $r - 1 - t_e < 0$, then we can still do better by moving off, as we hit here at time $2k + 2 + t_e$ (so all attacks that are catchable have been caught), so leave and get back to star at $*$ nodes at $3k + 4 + t_e$ in comparison to $k + m = 3k + 2 + r$ means an additional payoff depending on r's parity with t_e. In either case we get $\frac{r-t_e-1}{2(n+k)}$. But as this is negative it really means that all the attacks have already ended here, as $*$ attacks end at $k + m = 3k + 2 + r$ and as we arrive at $3k + 4 + t_e$ (and $r - 1 - t_e < 0$). Hence no additional values can be given and an overall value of $\frac{k+m-t-1}{2(n+k)}$ is given for this case.

But in both given scenario's it is still better than waiting and playing round all the $*$'s to end the game.

Hence it is at least as good to follow this strategy rather than repeatedly move between $*$ nodes. This means it is better than a single choice and therefore is the best thing to do in such a position.

Meaning the only option for starting at a $*$ node is to wait until $t = k$, the first real decision and decided to move to 1 getting a payoff of either $\frac{1}{2(n+k)} + \frac{m-1}{2(n+k)} = \frac{m}{2(n+k)}$

If we start at 1, then we can choose when to leave and visit $*$, say leave at time q and arrive at $*$ at $q + k + 2 \geq k$ as we know that from this position she will move back immediately (arriving back at $q + 2(k + 2)$ and hence will only get one of the nodes value of $\frac{1}{n+k}$, onece back at 1 we will travel back to $*$, as all attacks here are over (arriving at $q + 3(k + 2)$). This type of strategy will get her a payoff of

$$\frac{q+1}{2(k+1)} \times \frac{k+1}{n+k} + \frac{1}{k+1} + \frac{2(k+1)-(q+1)}{2(k+1)} \times \frac{k+1}{n+k} + \frac{(r-q-3)_+}{2(n+k)} = \frac{k+2+(r-q-3)_+}{n+k}$$

.

The other choice is not to visit in the middle and get all the attacks at 1, then move across, suggest $q = 2k+1$, now there is an opportunity to move to capture $*$ still occuring at $k + m = 3k + 2 + r$ when we arrive at $3k + 3$. (The complete sum depends on the parity of r) We will get $\frac{r}{2(n+k)}$. Meaning the overall payoff for playing this strategy is $\frac{k+1}{n+k} + \frac{r}{2(n+k)} = \frac{m}{2(n+k)}$.

In the case of $r - q - 3 \geq 0$ then we are comparing $\frac{k+r-q-1}{2(n+k)} < \frac{m}{2(n+k)}$

In the case of $r - q - 3 < 0$ then we are comparing $\frac{k+2}{2(n+k)} < \frac{m}{2(n+k)} = \frac{2k+2+r}{2(n+k)}$.

Hence the avaiable strategy to her if starting at 1 is to wait until $2k + 1$ and move to the $*$ nodes claiming $\frac{m}{2(n+k)}$. If starting at $*$ has to wait until k then move to 1 and then back to $*$ claiming a payoff of $\frac{m}{2(n+k)}$.

Hence the best she can do in this situation is to follow one of the strategies. □

It would be 'recommended' to follow the wait at 1 until time $2k + 1$ and then move as it is also optimal for all cases of m, whereas starting at $*$ is only valid for $m > 2(k + 1)$.

Altered proof below

Proof. We shall first consider the case of $m \leq 2(k+1)$. Consider starting at a $*$ node then the options for any time $t < k$ is wait (one time period and reconsider moving then) or move to another $*$ node or move to node 1.

Now immediately we can remove moving to another $*$ node as this is dominated by just waiting for two time periods. Now If waiting is the dominant strategy then we must continue to wait until time k upon which attacks at $*$ nodes begin.

Now consider being at a $*$ node for a time $k \leq t \leq k + m$ (when attacks are commencing), then the options are to move to another $*$ node claiming some benefit (if $t < k + m - 1$, otherwise no point in moving and infact the game is over at this point) or moving to 1 (arriving at $t + k + 2$) and catching some

attacks there hopefully (if $t < k + m - 1$, otherwise no point in moving and infact the game is over).

Note. The special case of $t = k$ in which she can wait for one time period will be covered later

Now consider that if moving to another $*$ node dominates moving to 1 then it will be done for all time (as the generated payoff is the same, for all time, apart from possibly, from time $t = k+m-2$ in which the generated payoff either way will be $\frac{1}{2(n+k)}$).

Now moving around the star nodes from time t until time $k + m - 1$ or $k + m$ (depending on the parity of these values i.e $t = 6$ $k + m - 1 = 10$ (even parity) or $t = 7$ $k + m = 11$ (odd parity)) gives us a payoff of exactly $\frac{1}{2(n+k)}$ per unit time, or more concretely

$$\frac{k+m-1-t}{2} \times \frac{1}{n+k} = \frac{k+m-1-t}{2(n+k)}$$

or

$$\frac{k+m-2-t}{2} \times \frac{1}{n+k} + \frac{1}{2(n+k)} = \frac{k+m-1-t}{2(n+k)}$$

In either case the same value is given (note. the initial payoff for being at a $*$ node at time t is not counted here)

The other decision to move to 1 and then make a decision, means arriving at 1 at time $t + k + 2 \geq 2k + 2 = 2(k + 1)$ meaning all attacks have already begun here (and infact it is impossible to decide to move back as return to $*$ at $t + 2(k + 2) \geq 3k + 4 > k + m$). This means that a payoff of

$$\frac{2k+m-(t+k+2)+1}{2(k+1)} \times \frac{k+1}{n+k} = \frac{k+m-1-t}{2(n+k)}$$

. Hence it is easy to see that it is not strictly dominating the alternative strategy.

For $t = k$ we can perform the addition strategy of wait one time period to gain $\frac{1}{2(n+k)}$ then we can decide to move to 1 getting $\frac{k+m-1-(k+1)}{2(n+k)}$, getting in total $\frac{m-1}{2(n+k)} = \frac{k+m-t-1}{2(n+k)}$, the same as above.

Now for starting at $*$ if it is before k then considering moving to 1 is dominated by just starting at 1, so the only option is to wait until k and then move to 1 (or move around 1 nodes, but they may not be enough) giving a total payoff of $\frac{m}{2(n+k)}$.

Now consider starting at node 1, we could consider waiting forever and getting $\frac{k+1}{2(n+k)}$, or we could consider moving to $*$ at some point in time say q.

Now doing the later will mean we arrive at $q + k + 2 \geq k$ (now assuming $q + k + 2 \leq k + m - 1$ i.e $q \leq m - 3$) then we will catch something as following the strategy for being at $*$ after time k. Meaning we get

$$\frac{q+1}{2(k+1)} \times \frac{k+1}{n+k} + \frac{1}{n+k} + \frac{k+m-1-(q+k+2)}{2(n+k)} = \frac{m}{2(n+k)}$$

(or worse if we don't arrive in time to catch things).

As $m \leq 2(k+1)$ it is clear that the option to wait at 1 and catch all attacks is better giving us in this case $\frac{k+1}{n+k}$.

Now for the case of $m > 2(k+1)$, let $m = 2(k+1) + r$ (where $r \geq 1$) $\square$

Lemma C.1. *The payoff for being at a $*$ node at a time $k \leq t \leq k+m-1$ is*

$$\frac{(k+m-t-1)_+}{2(n+k)}$$

As long as

Note. This payoff does not the intial payoff for being at $*$ at t, only future decisions

Proof. We will first cover $t \geq k+1$ (and later cover $t = k$ as a special case). Now the options are to go to another star arriving at $t+2$ (then remake a decision) or to move to 1.

Let us first look at moving to 1, then we arrive at 1 at time $t+k+2$ (and as $t \geq k$, it means all attacks occurring at 1 have begun) so we claim a payoff of

$$\frac{\min\{2(k+1)-x, (2k+m-(t+k+2)+1-x))_+\}}{2(k+1)} \times \frac{k+1}{n+k} = \frac{\min\{2(k+1), (k+m-t-1-x)_+\}}{2(n+k)}$$

Where $x \geq 0$ is the overlap with attacks already caught.

Now choosing to move to s $*$ nodes before moving to 1 gives a payoff of

$$s \times \frac{1}{n+k} + \frac{\min\{2(k+1)-(x)_+, \left(k+m-t-2s-1-(x-2s)_+\right)_+\}}{2(n+k)}$$

.

So it is best to move to all the $*$ nodes, which haven't been visited before, before moving to 1.

In the case of $m \leq 2(k+1)$, it is impossible to get any overlap as the time to leave and return is at least $2(k+2) > m$, so in this case $x = 0$. Also $(k+m-t-1)_+ \leq m-1 < 2(k+1)$, so the payoff from moving to 1 immediately becomes

$$\frac{k+m-t-1}{2(n+k)}$$

Similarly the other case becomes

$$s \times \frac{1}{n+k} + \frac{k+m-t-1-2s}{2(n+k)} = \frac{k+m-1-t}{2(n+k)}$$

In the case of $m > 2(k+1)$ (let $m = 2(k+1)+r$), overlap is definitely possible if 1 has been visited before. Now the game ends if she is at a $*$ at any time past $k+m = 3k+2+r$ or at 1 at any time past $2k+m = 4k+2+r$.

If 1 hasn't been visited before then $x = 0$ $\left(k + m - t - 2s - 1 - (x - 2s)_+\right)_+ = (k + m - t - 2s - 1)_+ = (3k + 1 + r - t - 2s)_+ \leq (2k + 1 + r - 2s)_+$,the payoff becomes either

$$s \times \frac{1}{n+k} + \frac{k+1}{n+k} = \frac{k+s+1}{n+k}$$

if $3k+1+r-t-2s > 2k+2$ (i.e $k-1+r-t-2s > 0$, so $2s < k+r-1-t$) or

$$s \times \frac{1}{n+k} + \frac{3k+1+r-t-2s}{2(n+k)} = \frac{k+m-t-1}{2(n+k)}$$

otherwise

Examining the first part gives us

$$\frac{k+s+1}{n+k} = \frac{2(k+1)+2s}{2(n+k)} < \frac{2(k+1)+k+r-1-t}{2(n+k)} < \frac{k+m-1-t}{2(n+k)}$$

. We will end at 1 at time $t+2s+k+2$, meaning we might as well choose $s = 0$ and arrive as early as possible getting $\frac{k+m-t-1}{2(n+k)}$

If there is some overlap then we will get the above if $x - 2s \leq 0$ (except the first part will be even worse with $-x$ in the numerator).

If $x > 2s$ though then we will get either

$$s \times \frac{1}{n+k} + \frac{2(k+1)-x}{2(n+k)} = \frac{2(k+1)+2s-x}{n+k}$$

if $3k + 1 + r - t - 2s - (x - 2s) > 2k + 2$ (i.e $k + 1 + r - t - x > 0$,

$$s \times \frac{1}{n+k} + \frac{3k+1-t-2s-(x-2s)}{2(n+k)} = \frac{3k+1-t-x+2s}{2(n+k)} = \frac{k+m-1-t-x+2s}{2(n+k)}$$

Hence in this case the best we can do is pick the highest s to try not to overlap as much.

In any case the best she can do from this position is $\frac{k+m-t-1}{2(n+k)}$ □

Proof. We will first deal with the case that $t \geq k + 1$. We will also first assume that there is no initial overlap of attacks at 1 caught, that is if we arrive at node 1 at $t + k + 2$ we will not be there at a time when attacks we previously caught would still be happening. Let the overlap be denoted by x, so first look at $x = 0$. Now our only choice from node $*$ is to move to s other $*$ nodes that we haven't yet visited and then move to 1 (in the hope of catching some attacks).

The payoff for doing this gives us

$$s \times \frac{1}{n+k} + \frac{\min\{2(k+1), (2k+m-(t+2s+k+2)+1-x)_+\}}{2(k+1)} \times \frac{k+1}{n+k}$$
$$= \frac{2s}{(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s-x)_+\}}{2(n+k)}$$

and as $x = 0$

$$\frac{2s}{2(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s)_+\}}{2(n+k)}$$

From this it should be clear that the payoff is non-decreasing in s and so choosing s as the maximum would seem to be a logical choice.

Now for a moment we will consider the future when we are at 1, as we will arrive at time $t+k+2+2s \geq 2k+2+2s \geq 2k+2$ all attacks occuring at 1 have been and we should no longer consider waiting or returning to this node, also moving away brings us back to $*$ nodes (arriving at time $t+2(k+2)+2s \geq 2k+4+2s$, when the attacks end at $k+m-1$ or $k+m$) , now as before all attacks have begun but they may not have ended. Hence we could consider moving around these $*$ nodes until the game ends. This means we actually get a payoff of

$$\frac{2s}{2(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s)_+\}}{2(n+k)} + \frac{\left(\frac{k+m-1-(t+2(k+2)+2s)}{2}\right)_+}{n+k}$$
$$= \frac{2s}{2(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s)_+\}}{2(n+k)} + \frac{(m-k-5-t-2s)_+}{2(n+k)}$$

or

$$\frac{2s}{2(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s)_+\}}{2(n+k)} + \frac{\left(\frac{k+m-2-(t+2(k+2)+2s)}{2}\right)_+}{n+k} + \frac{1}{2(n+k)}$$
$$= \frac{2s}{2(n+k)} + \frac{\min\{2(k+1), (k+m-t-1-2s)_+\}}{2(n+k)} + \frac{(m-k-5-t-2s)_+}{2(n+k)}$$

So we only need to worry about this is $m-k-5-t-2s > 0$. If $m = 2(k+1)+r$ then $m-k-5-t-2s = 2(k+1)+r-k-5-t-2s = k+r-3-t-2s$ So if $k+r-3-t-2s > 0$ then we will worry about this possibility of doing the movement at the ends. However if $k+r-3-t-2s > 0 \implies 3k+1+r-t-2s > 2(k+1) \implies k+m-t-1-2s > 2(k+1)$, meaing that the payoff actually becomes

$$\frac{2s}{2(n+k)} + \frac{2(k+1)}{2(n+k)} + \frac{m-k-5-t-2s}{2(n+k)} = \frac{k+m-t-3}{2(n+k)}$$

So in this case the choice of s is irrelevant, Hence we might as pick s to be the highest possible, call it $s_{max} = \min\{n-1-y, \frac{k+m-1-t}{2}\}$ (if odd parity) or $s_{max} = \min\{n-1-y, \frac{k+m-t}{2}\}$ (if even parity)(Note. In even parity we will get the starting payoff slightly differently). Where y is the number of currently visited $*$ nodes at time t.

For each type of parity let us cover the two cases

Odd Parity: 1. Let $n-1-y \geq \frac{k+m-1-t}{2}$ then the payoff we get becomes

$$\frac{\frac{k+m-1-t}{2}}{n+k} + \frac{\min\{2(k+1), \left(k+m-t-2\times\frac{k+m-1-t}{2}-1\right)_+\}}{2(n+k)}$$
$$= \frac{k+m-1-t}{2(n+k)} + \frac{\min\{2(k+1),0\}}{2(n+k)} = \frac{k+m-1-t}{2(n+k)}$$

2. Let $n-1-y < \frac{k+m-1-t}{2}$ then the payoff we get becomes

$$\frac{n-1-y}{n+k} + \frac{\min\{2(k+1), (k+m-t-2(n-1-y)-1)_+\}}{2(n+k)}$$

Further split into subcases

a) Let $k+m-t-2(n-1-y)-1 < 0$ then the payoff becomes

$$\frac{n-1-y}{n+k} < \frac{k+m-t-1}{2(n+k)}$$

b) Let $0 \leq k+m-t-2(n-1-y)-1 \leq 2(k+1)$ then the payoff becomes

$$\frac{n-1-y}{n+k} + \frac{k+m-t-2(n-1-y)-1}{2(n+k)} = \frac{k+m-t-1}{2(n+k)}$$

c) Let $k+m-t-2(n-1-y)-1 > 2(k+1)$ then the payoff becomes

$$\frac{n-1-y}{n+k} + \frac{2(k+1)}{2(n+k)} \quad = \frac{n+k-y}{n+k} < \frac{k+m-t-1}{2(n+k)}$$

As $k+m-t-2(n-1-y)-1 > 2(k+1) \implies k+m-t-1 > 2(n-1-y+k+1) = 2(n+k-y)$

Even Parity: 1. Let $n-1-y \geq \frac{k+m-t}{2}$ then the payoff we get becomes

$$\frac{\frac{k+m-t}{2}-1}{n+k} + + \frac{1}{2(n+k)} + \frac{\min\{2(k+1), \left(k+m-t-2\times\frac{k+m-t}{2}-1\right)_+\}}{2(n+k)}$$
$$= \frac{k+m-1-t}{2(n+k)} + \frac{\min\{2(k+1), (-1)_+\}}{2(n+k)} = \frac{k+m-1-t}{2(n+k)}$$

2. Let $n-1-y < \frac{k+m-t}{2}$ then the payoff we get becomes

$$\frac{n-1-y}{n+k} + \frac{\min\{2(k+1), (k+m-t-2(n-1-y)-1)_+\}}{2(n+k)}$$

Note. In this case, the 'problem' with having a final time only pick up $\frac{1}{2(n+k)}$ is not possible as we will have left before this time. Further split into subcases

a) Let $k+m-t-2(n-1-y)-1 < 0$ then the payoff becomes

$$\frac{n-1-y}{n+k} < \frac{k+m-t-1}{2(n+k)}$$

b) Let $0 \leq k+m-t-2(n-1-y)-1 \leq 2(k+1)$ then the payoff becomes

$$\frac{n-1-y}{n+k} + \frac{k+m-t-2(n-1-y)-1}{2(n+k)} = \frac{k+m-t-1}{2(n+k)}$$

c) Let $k+m-t-2(n-1-y)-1 > 2(k+1)$ then the payoff becomes

$$\frac{n-1-y}{n+k} + \frac{2(k+1)}{2(n+k)} \quad = \frac{n+k-y}{n+k} < \frac{k+m-t-1}{2(n+k)}$$

As $k+m-t-2(n-1-y)-1 > 2(k+1) \implies k+m-t-1 > 2(n-1-y+k+1) = 2(n+k-y)$

Now let us consider that there is some initial overlap, being more concretely $x = x(s) = (x(0)-2s)_+ = (x_0-2s)_+$. Then if $x-2s$ $leq0$ we get the same as above. Otherwise if $xgeq1$ then the payoff is

$$\frac{s}{n+k} + \frac{\min\{2(k+1), (k+m-t-1-2s-(x_0-2s))_+\}}{2(n+k)}$$
$$= \frac{s}{n+k} + \frac{\min\{2(k+1), (k+m-t-1-x_0)_+\}}{2(n+k)}$$

And again the payoff is non-decreasing in s, so again s_{max} is the best choice. We can follow a similar idea to the above when there was no overlap, though has the chance of being lower. Hence it will only harm the values

The exact algebra follows the same process expect now the second term in the visiting of node 1 has a subtraction and is therefore less rewarding.

Hence from this position the best the patroller can do is get a payoff of $\frac{k+m-t-1}{2(n+k)}$ and the best patroller strategy is to travel amongst as many $*$ nodes that have not already been visited and then head off to 1.

Now consider the special case of being at a $*$ at $t=k$, then the above can be followed to achieve no better than $\frac{m-1}{2(n+k)}$ or we can wait and pick up $\frac{1}{2(n+k)}$ by catching the second attack and then proceed to follow the above at $t=k+1$ getting $\frac{m-2}{2(n+k)}$. Either strategy yeilds the same payoff, so we shall suggest just strictly following the above. □

Theorem C.2 (Lower Bound)**.**

$$V \leq \max\left\{\frac{k+1}{n+k}, \frac{m}{2(n+k)}\right\}$$

Proof. Suppose the Patroller starts at a $*$ node at time $t \leq k-1$, then the options are to move to node 1 which is dominated by just starting at node 1 and waiting till $t+k+2$ (as no attacks are picked up at $*$ nodes. Or she can wait till time t and follow the above lemma getting a payoff of

$$\frac{1}{2(n+k)} + \frac{m-1}{2(n+k)} = \frac{m}{2(n+k)}$$

. So starting at $*$ provides a payoff of at most $\frac{m}{2(n+k)}$ Now consider starting at node 1 at time 0 then if we leave and want to return we can only do so once (as upon the first return the time will be at least $2(k+2)$ so all attacks will have begun).

Now split the solution into two regions for $m = 2(k+1)+r$, for $-2(k+1)+1 \leq r \leq 0$ and $r \geq 1$. Consider waiting until time q and then leaving, as you will hit a $*$ node at time $q+k+2$ meaning we can use the lemma to generate the max payoff here. For a total payoff of

1. If $q+k+2 > k+m = 3k+2+r$

$$\frac{q+1}{2(k+1)} \times \frac{k+1}{n+k} + \frac{(k+m-(q+k+2)-1)_+}{2(n+k)}$$
$$= \frac{q+1}{2(n+k)}$$

2. If $q+k+2 = k+m = 3k+2+r$

$$\frac{q+1}{2(k+1)} \times \frac{k+1}{n+k} + \frac{1}{2(n+k)} + \frac{(k+m-(q+k+2)-1)_+}{2(n+k)}$$
$$= \frac{q+2}{2(n+k)}$$

3. If $q+k+2 \leq k+m = 3k+2+r$

$$\frac{q+1}{2(k+1)} \times \frac{k+1}{n+k} + \frac{1}{n+k} + \frac{(k+m-(q+k+2)-1)_+}{2(n+k)}$$
$$= \frac{m}{2(n+k)}$$

So as these are non-decreasing in q we pick $q = 2k+1$. Then we get

1. If $3k+3 > 3k+2+r$ i.e $r \leq 0$

$$\frac{2k+1+1}{2(n+k)} = \frac{k+1}{n+k}$$

2. If $3k+3 = 3k+2+r$ i.e $r = 1$

$$\frac{2k+1+2}{2(n+k)} = \frac{2(k+1)+1}{2(n+k)} = \frac{m}{2(n+k)}$$

3. If $q+k+2 \leq k+m = 3k+2+r$ i.e $r \geq 2$

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

Hence if $m \leq 2(k+1)$ (i.e $r \leq 0$) then we get $\frac{k+1}{n+k}$ and if $m > 2(k+1)$ (i.e $r \geq 1$) we get $\frac{m}{2(n+k)}$ and hence the result. □

C.5 Proof of type-delayed attack

We begin with a lemma used in the proof

Lemma C.3. *When $k_{max} - i \leq t \leq k_{max} + i$ we get that*

$$\frac{(k_{max} + 1 + i - t)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} + \frac{(m - 2(i+1))_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} \geq \frac{(k_{max} - i + m - t - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)}$$

Proof. If $m \geq 2(i+1)$, then we get

$$\frac{k_{max} + 1 + i - t + m - 2(i-1)}{2\left(n + \sum_{j=1}^{h} k_j\right)} = \frac{k_{max} - i + m - t - 1}{2\left(n + \sum_{j=1}^{h} k_j\right)} \geq \frac{(k_{max} - i + m - t - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)}$$

If $m \leq 2(i+1)$, then we get

$$\frac{k_{max} + 1 + i - t}{2\left(n + \sum_{j=1}^{h} k_j\right)} \geq \frac{(k_{max} + 1 + i - t + m - 2(i+1))_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} = \frac{(k_{max} - i + m - t - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)}$$

□

The main proof of the attack lemma's future payoff

Note. The difference in future payoff depends on whether the patroller has the option to wait (till the last attack at the type i node) or whether they have no option but to move.

Remark. It is worth noting that the game essentially ends at time $t = 2k_{max}+m$, so this is really the last time we could possibly care about. Further as a type i node attacks finish at $t = k_{max} + i + m$, and so a movement to a type j node means arriving at $t = k_{max} + 2i + 2 + j + m \geq k_{max} + j + m \; \forall \, j$, meaning that no future attacks can be caught if at a type i node at $t = k_{max} + i + m$, and this means this is the artificial end for the game at a type i node.

Lemma C.4. *When the Type-delayed attack is possible, then the future payoff for being at a type i node at time t is given by:*

$$\begin{cases} \dfrac{(k_{max} + 1 + i - t)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} + \dfrac{(m - 2(i+1))_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} & \text{if } k_{max} - i \leq t \leq k_{max} + i \\ \dfrac{(k_{max} - i + m - t - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} & \text{if } t \geq k_{max} + i + 1 \end{cases}$$

Proof. We will first prove the statement for when $t \geq k_{max} + i + 1$, by the use of strong backwards induction.

Base Case:
For $t = k_{max} + i + m$, as above it is known that no future attacks can be caught (and the game is over) as arriving at a type j means arriving at $k_{max} + 2i + j + 2 + m \geq k_{max} + j + m$ meaning all attacks are over ($\forall\, j$). Hence the future payoff is 0. Now the formula gives

$$\frac{(k_{max} - i + m - (k_{max} + i + m) - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} = \frac{(-2i - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)} = 0$$

Hence it is true for $t = k_{max} + i + m$.

Induction hypothesis:
For some $t_1 \geq k_{max} + i + 1$ assume that the formula for $t \geq k_{max} + i + 1$ is true for all $t \geq t_1 + 1$.

Induction step:
At a type i node at time $t_1 \geq k_{max} + i + 1$, all attacks have begun at this node so waiting is not really an option (i.e it is clearly dominated by moving immediately). So moving to a type j means arriving at $t_1 + i + 2 + j$ giving an immediate payoff and meaning (under the Induction hypothesis) a future payoff is received. Giving at best a total payoff of

$$\underbrace{\frac{\min(2(j+1), (k_{max} + j + m - (t_1 + i + 2 + j) + 1)_+)}{2(j+1)} \times \frac{j+1}{n + \sum_{j=1}^{h} k_j}}_{\text{Immediate reward at type } j}$$
$$+ \underbrace{\frac{(k_{max} - j + m - (t_1 + i + 2 + j) - 1)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)}}_{\text{Future reward from type } j}$$
$$= \frac{\min(2(j+1), (k_{max} + m - t_1 - i - 1)_+)}{2\left(n + \sum_{j=1}^{h} k_j\right)} + \frac{(k_{max} - i + m - t_1 - 2j - 3)_+}{2\left(n + \sum_{j=1}^{h} k_j\right)}$$

Now we will split it into the subcase of $2(j+1) \geq k_{max} + m - t_1 - i - 1$. This means $k_{max} + m - t_1 - 3 - 2j \leq 0$, then the payoff becomes

$$\frac{(k_{max} + m - t_1 - i - 1)_+)}{2\left(n + \sum_{j=1}^{h} k_j\right)}$$

Otherwise if $2(j+1) < k_{max} + m - t_1 - i - 1$. This means $k_{max} + m - t_1 - i - 3 - 2j >$

0, then the payoff becomes

$$\frac{2(j+1)+k_{max}-i+m-t_1-2j-3}{2\left(n+\sum_{j=1}^{h}k_j\right)}=\frac{k_{max}-i+m-t_1-1}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$
$$=\frac{(k_{max}-i+m-t_1-1)_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$

Hence by mathetmatical induction it is true for $t \geq k_{max}+i+1$ and the form is correct.

Next we seek to prove the formula for the values of $t \leq k_{max}+i$, again using strong backwards induction.

Base case:
For $t = k_{max}+i$, we have the option to wait for one unit of time (to catch the final attack at this node) and then claim the reward for this and the future rewards, or can move immediately to some type j node (arriving at $k_{max}+2i+2+j$ and claim the reward and future rewards.

Waiting gives

$$\frac{1}{2(i+1)}\times\frac{i+1}{n+\sum_{j=1}^{h}k_j}+\frac{(k_{max}-i+m-(k_{max}+i+1)-1)_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$
$$=\frac{1}{2\left(n+\sum_{j=1}^{h}k_j\right)}+\frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$

Moving gives

$$\frac{\min(2(j+1),(k_{max}+j+m-(k_{max}+2i+2+j)+1)_+)}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$
$$+\frac{(k_{max}-j+m-(k_{max}+2i+2+j)-1)_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$
$$=\frac{\min(2(j+1),(m-2i-1)_+)+(m-2i-2j-3)_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$$

When moving it will depend on if $2(j+1) \geq m-2i-1$, which means $m-2i-2j-3 \leq 0$ giving $\frac{(m-2i-1)_+}{2\left(n+\sum_{j=1}^{h}k_j\right)}$

Or if $2(j+1) < m-2i-1$, meaning $m-2i-2j-3 > 0$ gives $\frac{2(j+1)+m-2i-2j-3}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{m-2i-1}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{1}{2\left(n+\sum_{j=1}^{h} k_j\right)} + \frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$ (Note. Here $m-2(i+1) > 1$)

Hence with either option the best that the patroller can do is $\frac{1}{2\left(n+\sum_{j=1}^{h} k_j\right)} + \frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$, which is the correct form of the formula.

Induction hypothesis:
Assume that the formula for $k_{max} - i \leq t \leq k_{max} + i$ is true for all $t \geq t_1 + 1$.

Induction step:
At a type i node at time $k_{max} - i \leq t_1 \leq k_{max} + i - 1$, the options is to wait for some period of time, say q periods, and then move to a type j node and claim future rewards (which will depends on which case the arrival time falls into), this decision means we will arrive at the type j node at $t_1+q+i+2+j \geq k_{max}+q+2+j$ meaning we will always fall into the second category for future rewards.

This means the future payoff for such a decision will be

$$\frac{q}{2(i+1)} \times \frac{i+1}{n+\sum_{j=1}^{h} k_j} + \frac{\min(m, 2(j+1), (k_{max}+j+m-(t_1+q+i+2+j)+1)_+)}{2(j+1)} \times \frac{j+1}{n+\sum_{j=1}^{h} k_j}$$
$$+ \frac{(k_{max}-i+m-(t_1+i+2+j)-1)_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$
$$= \frac{q+\min(2(j+1), (k_{max}-i+m-t_1-1-q)_+) + (k_{max}-j+m-(t_1+q+i+2+j)-1)_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$
$$= \frac{q+\min(m, 2(j+1), (k_{max}-i+m-t_1-1-q)_+) + (k_{max}-i-2j+m-t_1-3-q)_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$
$$= \frac{q+\min(m, 2(j+1), (k_{max}-i+m-t_1-1-q)_+) + (k_{max}-i+m-t_1-1-q-2(j+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$

Now if $m \geq 2(j+1) \geq k_{max} - i + m - t_1 - 1 - q \geq 0$ then we get

$$\frac{q+k_{max}-i+m-t_1-1-q}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{k_{max}-i+m-t_1-1}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$

If $m \geq 2(j+1) \geq k_{max} - i + m - t_1 - 1 - q$ and $k_{max} - i + m - t_1 - 1 - q \leq 0$

then we get $\frac{q}{2\left(n+\sum_{j=1}^{h} k_j\right)}$

If $m \geq k_{max}-i+m-t_1-1-q \geq 2(j+1)$ then we get $\frac{q+2(j+1)+k_{max}-i+m-t_1-1-q-2(j+1)}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{k_{max}-i+m-t_1-1}{2\left(n+\sum_{j=1}^{h} k_j\right)}$

If $k_{max}-i+m-t_1-1-q \geq m \geq 2(j+1)$ then we get $\frac{q+2(j+1)+k_{max}-i+m-t_1-1-q-2(j+1)}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{k_{max}-i+m-t_1-1}{2\left(n+\sum_{j=1}^{h} k_j\right)}$

If $k_{max}-i+m-t_1-1-q \geq 2(j+1) \geq m$ then we get $\frac{q+m+k_{max}-i+m-t_1-1-q-2(j+1)}{2\left(n+\sum_{j=1}^{h} k_j\right)} =\leq \frac{q+2(j+1)+k_{max}-i+m-t_1-1-q-2(j+1)}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{k_{max}-i+m-t_1-1}{2\left(n+\sum_{j=1}^{h} k_j\right)}$

In every case we know that the value is less than $\frac{(k_{max}+1+i-t)_+}{2\left(n+\sum_{j=1}^{h} k_j\right)} + \frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$.

Hence by mathetmatical induction it is true for $k_{max} - i \leq t \leq k_{max} + i$ and the form is correct.

□

The theorem's proof is now easy to follow

Proof. Starting at a type i node means waiting till time $t = k_{max} - i$ and then recieving a payoff from the previous lemma. This should be clear as for any time, $s < k_{max} - i$, the decision is to wait or move to another type node and as no attacks can be claimed by waiting if moving was better then the patroller might as well start there. Hence starting at type i provides a payoff of

$$\frac{1}{2(i+1)} \times \frac{i+1}{n+\sum_{j=1}^{h} k_j} + \frac{2i+1}{2\left(n+\sum_{j=1}^{h} k_j\right)} + \frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)} = \frac{i+1}{n+\sum_{j=1}^{h} k_j} + \frac{(m-2(i+1))_+}{2\left(n+\sum_{j=1}^{h} k_j\right)}$$

So if $m \geq 2(i+1)$ then we get $\frac{m}{2\left(n+\sum_{j=1}^{h} k_j\right)}$ and if $m < 2(i+1)$ then we get $\frac{i+1}{n+\sum_{j=1}^{h} k_j}$. Hence if $m \geq 2(k_{max}+1)$ then we get $\frac{m}{2\left(n+\sum_{j=1}^{h} k_j\right)}$ and if $m < 2(k_{max})$ we get $\frac{k+1}{n+\sum_{j=1}^{h} k_j}$. □

C.6 Generalised ARHP

As an extension to the Alternating Random Hamiltonian Patrol(ARHP) is the Block Random Hamiltonian Patrol(BRHP).

Definition C.5 (Block Random Hamiltonian Patrol(BRHP))**.** A k type *Block Random Hamiltonian Patrol* (*BRHP*) is a mixed strategy following the Hamiltonian Patrol but with a probability p_i of starting at a "type i" node for $i = 1, .., k$, where $\sum_{i=1}^{k} p_i = \frac{k}{n}$.

Lemma C.6. *When n and m are both multiples of k, then following the k type Block 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 length $m = km'$, then $W(I)$ contains m' "type i" nodes for $i = 1, ..., k$. Therefore by following the k type BRHP, $\boldsymbol{\pi}_{BRHP}$, with probability p_i at "type i" for $i = 1, ..., k$. Then

$$P(\boldsymbol{\pi}_{BRHP}, [i, I]) \geq \underbrace{\overbrace{p_1}^{\text{type 1}} + ... + \overbrace{p_k}^{\text{type } k} + p_1 + ... + p_n + ... + ... + p_1 + ... + p_k = 2m'}_{\text{m=km'elements}} =$$

$$m'p_1 + ... + m'p_k = m'\frac{k}{n} = \frac{m}{n} \quad \forall i \in N \quad \forall I \subseteq \mathcal{T}$$

Hence as it holds for all pure attacks

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

Hence $V \geq \frac{m}{n}$. □

Note. In the one-off game, the BRHP strategy is always feasible. However in the periodic game, the BRHP strategy is only feasible if $T = k'n$ for some $k' \in \mathbb{N}$.

D Improving random oscillations

D.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 $2(n+k)$.

D.2 Naive improvement 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 $V \geq \frac{2m}{2(n+k)+mn}$

D.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

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

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

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

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

D.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

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

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.

E Optimal Solution for a Random Attacker

F Optimal Solution for Local-Observations