A graph Patrol Problem with a random attacker and an observable patroller

Thomas Lowbridge
School of Mathematical Sciences
University of Nottingham

March 21, 2018

Contents

1 Introduction to a random attacker patroller game with observation

The model has a graph, $Q = (N, E)$, with a set of nodes labeled 1 to n, $N = \{1, ..., n\}$, and a set of edges linking these nodes. The adjacency matrix $a = (a_{i,j})_{i,j \in N}$, has $a_{i,j} = 1$ if i and j are adjacent and $a_{i,j} = 0$ if they are not adjacent. By definition we will use $a_{i,i} = 1 \quad \forall i \in N$.

An attacker has some attack time for node i, called X_i and chooses to attack node i with some probability, p_i. The attackers arrive according to some Poisson process with rate Λ, so by Poisson thinning they arrive at node i according to a Poisson process with rate $\lambda_i = \Lambda p_i$.

The patroller, uses some walk (with possible waiting) to patrol the graph.We assume that a patrollers walk is able to capture all attacks that have already begun, but not completed. But unlike the 'normal' setting the past unit time, the attackers do not start their attacks and instead will wait for the patroller to leave. Each missed attack at node i inccures a cost of c_i to the patroller.

We can formulate the state space, as the delineation of separate nodes. $\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) = \quad | \quad s_i = 1, 2, ..., v_i = 0, 1, 2, ... \quad \forall i \in N\}$. Where $\boldsymbol{s} = (s_1, ..., s_n)$ has each s_i represent the number of time periods since the last visit for that node i and $\boldsymbol{v} = (v_1, ..., v_n)$ has each v_i represent the number of attackers present in the last time period when the node i was last visited (i.e The number of attackers known to be beginning their attack s_i time ago at node i).

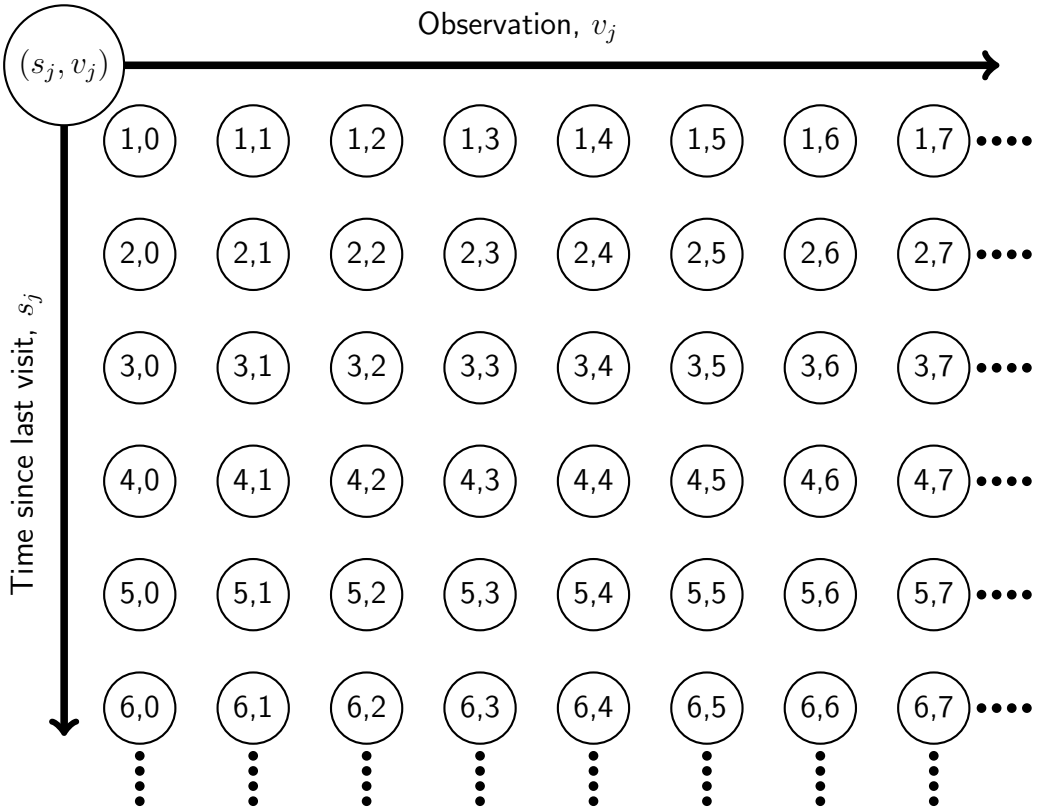

Figure 1.1: State space diagram

The s_i increment by 1 if the node is not visited upon each action, or if the node is visited reset to $s_i = 1$. The v_i do not change for nodes not visited, when a node is visited, the v_i 'reset' according to the Poisson distribution $Po(\lambda_i \times 1) = Po(\lambda)$. Due to $s_i = 1$ if and only if the patroller is currently at this node, we will use $l(\boldsymbol{s}) = \arg\min_{i \in N} s_i$ to represent the current node.

As the future of the process is independent of its past, the process can be formulated as a Markov decision process(MDP), where at the end of the period, the patroller chooses which adjacent node to visit. Thus the action space is $\mathcal{A} = \{j \,|\, a_{l(\boldsymbol{s}),j} = 1\}$, with a deterministic, stationary policy, $\pi : \Omega \to \mathcal{A}$.

The transitions of the MDP aren't entirely deterministic, $\boldsymbol{s}$ is purely deterministic, but $\boldsymbol{v}$ is partially probabilistic. In state $(\boldsymbol{s}, \boldsymbol{v})$ with the decision to visit node $i \in \mathcal{A}$, then the state will transition to $(\widetilde{\boldsymbol{s}}, \widetilde{\boldsymbol{v}})$ where $\widetilde{s}_j = s_j + 1$ if $j \neq i$ and $\widetilde{s}_j = 1$ if $j = i$ and $\widetilde{v}_j = v_j$ if $j \neq i$ and $v_j \sim Po(\lambda)$ if $j = i$.

To write down the cost function, which is dependent on the state $(\boldsymbol{s}, \boldsymbol{v})$ and the action to visit node i chosen, we will look at the expected cost of incurred at all nodes and sum these costs for the next time period.

1.1 Derivation of the Cost function

We will now work under the assumption that the patroller catches ongoing attacks at the end of the time period, but attacks arriving after their decision has been made to move to a node will be stopped and become the observed value.

We have two parts to the cost, the attacks which arrive and complete in the next time period and the cost of the locally-observed attacks that will finish.

First we will deal with the cost of attacks which arrive and complete. Then the cost for a node which is not being chosen to be visited we will get the 'old' cost from the prior model i.e can arrive between 1 and $s_j + 1$ and finish in this next time period

This gives a value of

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

However the cost for a node which is being chosen to be visited, will only have arrivals between 1 and s_j able to finish in this next time period (as the ones between s_j and $s_j + 1$ are 'stopped' and become locally-observed attackers)

This gives a value of

$$\begin{aligned}
c_j\lambda_j \int_0^{s_j-1} P(t < X_j \leq t+1)dt &= c_j\lambda_j \int_1^{s_j} P(u-1 < X_j \leq u)du \\
&= c_j\lambda_j(\underbrace{\int_0^{s_j} P(u-1 < X_j \leq u)du}_{\text{Old amount}} - \underbrace{\int_0^{1} P(u-1 < X_j \leq u)du}_{\text{Losing the last period arriving and finishing}}) \\
&= c_j\lambda_j(\int_{s_j-1}^{s_j} P(X_j \leq u)du - \int_0^{1} P(X_j \leq u)du) \\
&= c_j\lambda_j(\int_{s_j-1}^{s_j} P(X_j \leq t)dt - \int_0^{1} P(X_j \leq t)dt)
\end{aligned}$$

Now the contribution for the know local-observed attackers is just the expected number of them that finish in this time period that is their contribution is given by

$$c_j v_j P(s_j - 1 < X_j \leq s_j)$$

Giving an overall cost contribution from node j when the action to move to node i is chosen in current state $(\boldsymbol{s}, \boldsymbol{v})$.

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = \begin{cases} c_j(\lambda_j \int_{s_j-1}^{s_j} P(X_j \leq t)dt + v_j P(s_j - 1 < X_j \leq s_j)) \text{ for } i \neq j \\ c_j(\lambda_j(\int_{s_j-1}^{s_j} P(X_j \leq t)dt - \int_0^1 P(X_j \leq t)dt) + v_j P(s_j - 1 < X_j \leq s_j)) \text{ for } i = j \end{cases} \tag{1}$$

That is the only difference between having chosen the nodes is the effect on the cost is to remove the one time period closest to when we visit.

With $C(\boldsymbol{s}, \boldsymbol{v}, i) = \sum_{j=1}^{n} C_j(\boldsymbol{s}, \boldsymbol{v}, i)$ being the cost function for the MDP.

1.2 Alternative Cost function for instantly moving patroller

We will now derive and alternative cost function under the assumption that the patroller moves instantly and collects attacks throughout the window.

The cost function when we do not choose to visit the node remains the same, however we need to slightly alter the cost function for the node we are choosing to visit. The incurred cost in the next time period is 0 as we move instantly and catch all attack that would end up here.

Hence we would have a cost function of

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = \begin{cases} c_j(\lambda_j \int_{s_j-1}^{s_j} P(X_j \leq t)dt + v_j P(s_j - 1 < X_j \leq s_j)) \text{ for } i \neq j \\ 0 \text{ for } i = j \end{cases} \tag{2}$$

Note. The bounding reduction that occurs in the next section still holds with this cost function. I.e. the state space is reduced the same whichever cost function we pick.

1.3 Bounding Problem

We will now make the assumptions that X_j is bounded and we will let

$$B_j = \min\{k \in \mathbb{N}_0 \,|\, P(X_j \leq k) = 1\}$$

. I.e B_j is the least integer such that it is an upper bound for X_j. (we can think of this as the ceiling function on the real bound of X_j i.e if $X_j \leq RB_j$, where $RB_j \in \mathbb{R}$ then $B_j = \lceil RB_j \rceil$). The cost function is the same for all $s_j \geq B_j + 1$ so we will restrict our state space to $s_j \leq B_j + 1$.

We will also limit the observation state space, from $Po(\lambda)$ for the observation transition and placing a bound on this Poisson distribution, named b_j, so we are now drawing from a truncated Poisson distribution, henceforth called $TPo(\lambda, b_j)$. Then we can immediately say that the $v_j \leq b_j$ state is finite.

So our modified transition is $\widetilde{s_j} = \min(s_j + 1, B_j + 1)$ if $j \neq i$ and $\widetilde{s_j} = 1$ if $i = j$. $\widetilde{v_j} = v_j$ if $i \neq j$ and $v_j \sim TPo(\lambda, b_j)$ if $i = j$.

The truncated Poisson distribution, $TPo(\lambda, b)$ acts like normal Poisson for any value not equal to b, with the tails probability stored at b. That is $P(TPo(\lambda, b) = i) = \begin{cases} P(Po(\lambda) = i) \text{ If } i \neq b \\ P(Po(\lambda) \geq b) \text{ If } i = b \\ 0 \text{ Otherwise} \end{cases}$

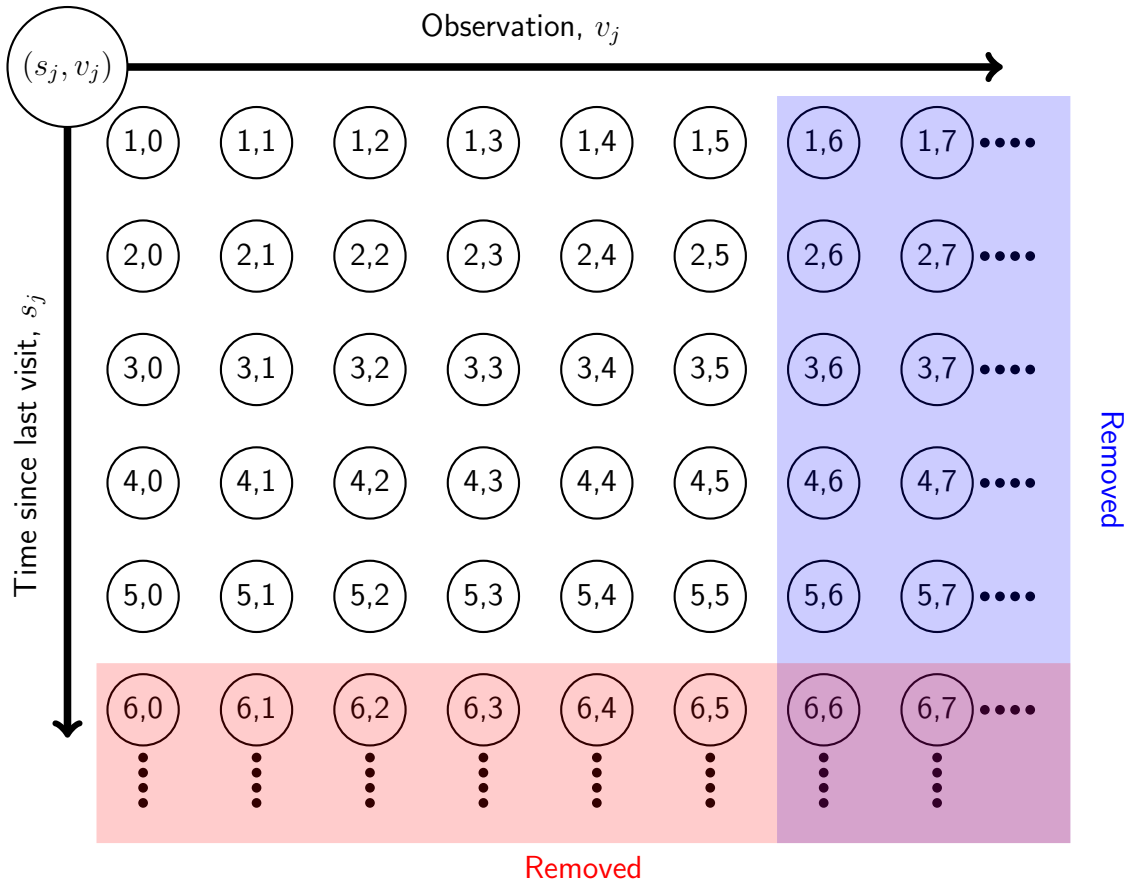

Figure 1.2: State space diagram, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Further reduction is possible as if $X_j \leq B_j$ then any observations v_j which started $s_j - 1$ time units ago (note that start at the end of a time period) is bound to have finished if $X_j \leq s_j - 1$ and as $X_j \leq B_j$ it means $s_j \geq B_j + 1$ implies all the waiting attacks have completed. So our 'final' $s_j = B_j + 1$ must have $v_j = 0$.

Example 1.1. Say We have $X_j \sim U(0, 3.7)$ then we have that $B_j = 4$ and then as the cost function is the same for all $s_j \geq 4 + 1 = 5$ as we get

$$C_j(s_j, v_j, i) = c_j \lambda_j + 0 \quad \forall j$$

Also note at $s_j = 5$ we have that $v_j = 0$ as the attacks have had 4 time periods respectively to complete and as at maximum it takes 3.7 they must be complete.

So our new state space is further reduced to having only $(B_j + 1, 0)$ for node each node j.

So $\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) | s_i = 1, 2, .., B_i + 1, v_i = 0, 1, ..., b_i, \text{ if } s_i \in \{B_i + 1\} \text{ then } v_i = 0 \, \forall i \in N\}$

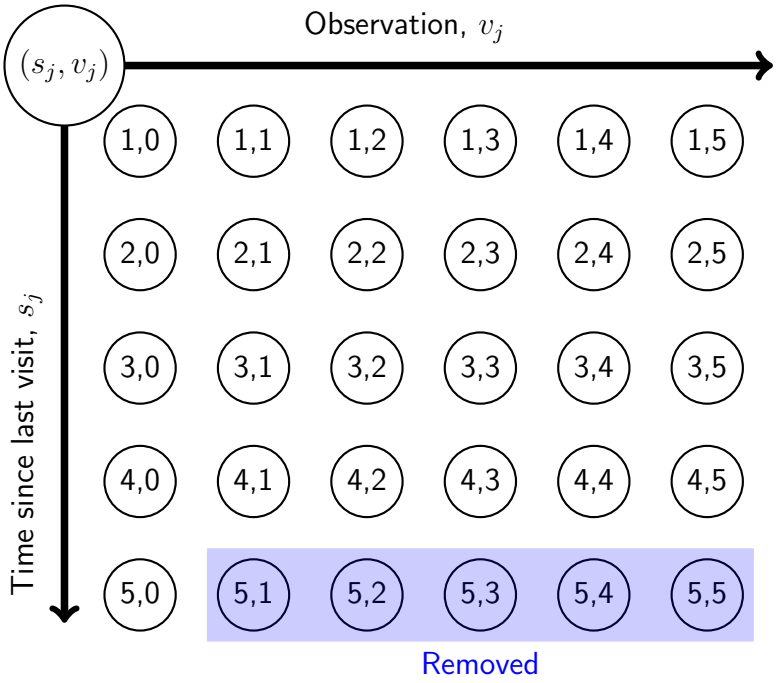

Figure 1.3: State space diagram, with $b_j = 5$ and $B_j = 4$ and further reduction of states with $s_j = 5$ having to have $v_j = 0$

With further modified transitions that if $\widetilde{s_j} = B_j + 1$ then $\widetilde{v_j} = 0$ for $i \neq j$ (Or equivalently if $s_j = B_j$ then $\widetilde{v_j} = 0$ for $i \neq j$).

We will define $\phi_s(\boldsymbol{s}, i) = \widetilde{\boldsymbol{s}}$, i.e the evolved s states and similarly $\phi_v(\boldsymbol{s}, \boldsymbol{v}, i) = \widetilde{\boldsymbol{v}}$ i.e the evolved v states.

Note. We note that the evolution of the v states depends on the s states, as if the $s_j = B_j$ then evolved state of v_j becomes 0.

1.4 Problem formulation

The objective of this MDP is to minimize the total long-run cost rate among the n nodes of the graph. Our state space is finite because the attack time and local-observations are bounded above. The action space is finite because the number of nodes is finite. Hence we only consider deterministic , stationary policies (i.e we will always take the same action in the same state and time is irrelevant).

Unfortunately the state transitions are not fully deterministic, while the s states evolve this way, the v state which is visited will 'reset' to a random number (drawn from our truncated poisson). We will define $\psi_\pi(\boldsymbol{s}, \boldsymbol{v}) = (\psi_{\pi,s}(\boldsymbol{s}, \boldsymbol{v}), \psi_{\pi,v}(\boldsymbol{s}, \boldsymbol{v})) = (\phi_s(\boldsymbol{s}, \pi(\boldsymbol{s}, \boldsymbol{v})), \boldsymbol{v}), \phi_v(\boldsymbol{s}, \boldsymbol{v}, \pi(\boldsymbol{s}, \boldsymbol{v})))$.

That is $\psi_{\pi,s}(\boldsymbol{s}, \boldsymbol{v})$ is purely deterministic, but $\psi_{\pi,v}(\boldsymbol{s}, \boldsymbol{v})$ is partially random.

As the state space is finite and π is a deterministic ,stationary policy, eventually we will visit the same state again. This should hold even though we are partially probabilistic. However the process will not regenerate itself, even though the same action will be taken we may end up in a different state state.

However for an initial state $(\boldsymbol{s}_0, \boldsymbol{v}_0)$ the policy π does induce an indefinite,sequence of states $\{\psi_\pi^k(\boldsymbol{s}_0, \boldsymbol{v}_0) \, | k = 0, 1, 2...\}$ where $\psi_\pi^k(\boldsymbol{s}_0, \boldsymbol{v}_0)$ is the evolved state starting from the initial state after we apply π k times to the resulting state.

Note. We will have 'cycles' of states which do repeat in the process, but we may jump to different cycles, due to the probabilistic nature of the evolved observations. This means cycles can be of different lengths.

That is to say we follow a selection of patrol patterns each for a different period of time. This selection depends on the probabilistic evolution.

Therefore, if we apply the policy π to an initial state $(\boldsymbol{s}_0, \boldsymbol{v}_0)$ we can write the long-run cost rate at node i as

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

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

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

Where Pi is the class of deterministic, stationary patrol policies.

Note. A technical note is that we use the minimum rather than the infimum as Pi is finite

If we take C^{OPT} and divide it by the overall arrival rate, Λ then we get the minimized long-run cost incurred for each attack. Furthermore if we set $c_i = 1\,\forall i \in N$, then this ratio can be interpreted as the probability of not detecting an attack, i.e the evasion probability.

We will also note that while $V_i(\pi, \boldsymbol{s}_0, \boldsymbol{v}_0)$ depends on initial state $(\boldsymbol{s}_0, \boldsymbol{v}_0)$, the optimal cost rate $C^{\mathrm{OPT}}(\boldsymbol{s}_0, \boldsymbol{v}_0)$ does not if the graph is connected. This is because from any initial state, we can construct a policy π which produces our optimal selection of patrol patterns.

Hence we will drop the dependence on the initial state and the value of C^{OPT} will be the same for every initial state.

2 Problem Relaxation

We know look at relaxing our problem to easier to solve problems.

2.1 Multiple Node problem

We will now relax our class of problems to allow ourselves to visit multiple nodes in one time period, as long as the overall long-run visit rate is not greater than 1. We will call this Problem the *multiple node* (MN) problem. To extend ourselves to this problem we will create a new class of polices, Π^{MN} which will allow the patroller to visit any set of nodes during the next timer period.

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

Where $\alpha_i = 1$ if the patroller will visit node i in the next time period and $\alpha_i = 0$ if the patroller will not visit node i in the next time period. This patrol policy is more complex then those previously discussed and $\Pi \subset \Pi^{\mathrm{MN}}$.

We now define the long-run rate at which the patroller visits node i, following π

$$\mu_i(\pi, \boldsymbol{s}_0, \boldsymbol{v}_0) = \lim_{N\to\infty} \frac{1}{N} \sum_{k=0}^{N-1} \alpha_{\psi_\pi^k(\boldsymbol{s}_0, \boldsymbol{v}_0), i}$$

We now impose the *total-rate* constraint, that is the long-run overall visit rate to all nodes is no greater than 1.

$$\sum_{i=1}^{n} \mu_i(\pi, \boldsymbol{s}_0, \boldsymbol{v}_0) \quad \forall (\boldsymbol{s}_0, \boldsymbol{v}_0) \in \Omega$$

and we denote the set of policies, which satisfy this constraint by

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

Again $\Pi \subset \Pi^{\mathrm{TR}}$

As discussed previously the optimal long-run cost rate does not depend on the initial state so we will drop the notation.

Therefore our relaxed problem can be formulated as

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

Hence due to this class of policies including the old class of policies, it is only possible to get a lower optimal value and hence $C^{\mathrm{OPT}} \geq C^{\mathrm{TR}}$.

2.2 Lagrangian relaxation

We now relax the problem again by incorporating the total rate constraint into the objective function, with a Lagrange multiplier, $\omega \geq 0$. This forms

$$\begin{aligned} C(\omega) &= \min_{\pi \in \Pi^{\mathrm{MN}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + \omega \left(\sum_{i=0}^{n} \mu_i(\pi) - 1 \right) \right\} \\ &= \min_{\pi \in \Pi^{\mathrm{MN}}} \sum_{i=1}^{n} (V_i(\pi) + \omega \mu_i(\pi)) - \omega \end{aligned}$$

By incorporating the total-rate constraint as a Lagrange multiplier we can drop the constraint, so that the patroller can choose to visit any number of nodes in each time period (admittedly costing ω).

We have that for any $\omega \geq 0$

$$\begin{aligned} C^{\mathrm{TR}} = \min_{\pi \in \Pi^{\mathrm{TR}}} \sum_{i=1}^{n} V_i(\pi) &\geq \min_{\pi \in \Pi^{\mathrm{TR}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + \omega \left(\sum_{i=0}^{n} \mu_i(\pi) - 1 \right) \right\} \\ &\geq \min_{\pi \in \Pi^{\mathrm{MN}}} \left\{ \sum_{i=1}^{n} V_i(\pi) + \omega \left(\sum_{i=0}^{n} \mu_i(\pi) - 1 \right) \right\} = C(\omega) \end{aligned}$$

The first inequality follows as the total-rate constraint is obeyed in Π^{TR} and hence the $\omega \left(\sum_{i=0}^{n} \mu_i(\pi) - 1 \right) \leq 0$ and the second holds, as the total rate constraint is dropped and we have got a bigger state space, so we can only do better and hence possibly get a lower value.

Hence we have a string of inequalities $C^{\mathrm{OPT}} \geq C^{\mathrm{TR}} \geq C(\omega)$.

To solve the problem of finding $C(\omega)$ we can note that the problem can be split up into each node, for each node i we want to minimize $V_i(\pi) + \omega \mu_i(\pi)$, where ω can be interpreted as the service charge when the patroller visits the node.

2.3 Single node problem

We will now focus on the problem of having a single node where we pay a service cost of ω to visit it. Namely we solve the objective function for $C(\omega)$ with the subscript for the node i removed

$$\min_{\pi \in \Pi^{\text{MN}}} V(\pi) + \omega\mu(\pi)$$

That is we seek to minimize the single nodes long-run average cost, when we have to pay a service charge of ω to visit the node.

3 Deterministic Attack time

Consider the case where $X_j = x_j$, where x_j is a constant (So $B_j = \lceil x_j \rceil$). So note that

$$P(X_j \leq a) = \begin{cases} 1 \text{ if } a \geq x_j \\ 0 \text{ if } a < x_j \end{cases}$$

Then our cost function becomes

For $s_j < B_j$

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = 0$$

For $s_j = B_j$

$$\begin{aligned} C_j(\boldsymbol{s}, \boldsymbol{v}, i) &= \begin{cases} c_j\lambda_j \int_{B_j-1}^{B_j} P(X_j \leq t)dt + c_j v_j P(B_j - 1 < X_j \leq B_j) \text{ for } i \neq j \\ c_j\lambda_j (\int_{B_j-1}^{B_j} P(X_j \leq t)dt - \int_0^1 P(X_j \leq t)dt) + c_j v_j P(B_j - 1 < X_j \leq B_j) \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j \int_{x_j}^{B_j} 1dt + c_j v_j \times 1 \text{ for } i \neq j \\ c_j\lambda_j (\int_{x_j}^{B_j} 1dt - \int_0^1 P(X_j \leq t)dt) + c_j v_j \times 1 \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j (B_j - x_j) + c_j v_j \text{ for } i \neq j \\ c_j\lambda_j ((B_j - x_j) - \int_0^1 P(X_j \leq t)dt) + c_j v_j \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j R_j + c_j v_j \text{ for } i \neq j \\ c_j\lambda_j (R_j - \int_0^1 P(X_j \leq t)dt) + c_j v_j \text{ for } i = j \end{cases} \end{aligned}$$

Where we define $R_j = B_j - x_j$.

Note. In the case of $i = j$: If $x_j > 1$ then our function becomes $c_j\lambda_j R_j + c_j v_j$. If $x_j \leq 1$ then our function becomes $c_j v_j$

For $s_j = B_j + 1$

$$\begin{aligned} C_j(\boldsymbol{s}, \boldsymbol{v}, i) &= \begin{cases} c_j\lambda_j \int_{B_j}^{B_j+1} P(X_j \leq t)dt + c_j v_j P(B_j < X_j \leq B_j + 1) \text{ for } i \neq j \\ c_j\lambda_j (\int_{B_j}^{B_j+1} P(X_j \leq t)dt - \int_0^1 P(X_j \leq t)dt) + c_j v_j P(B_j < X_j \leq B_j + 1) \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j \int_{B_j}^{B_j+1} 1dt + c_j v_j \times 0 \text{ for } i \neq j \\ c_j\lambda_j (\int_{B_j}^{B_j+1} 1dt - \int_0^1 P(X_j \leq t)dt) + c_j v_j \times 0 \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j \times 1 \text{ for } i \neq j \\ c_j\lambda_j (1 - \int_0^1 P(X_j \leq t)dt) \text{ for } i = j \end{cases} \\ &= \begin{cases} c_j\lambda_j \text{ for } i \neq j \\ c_j\lambda_j (1 - \int_0^1 P(X_j \leq t)dt) \text{ for } i = j \end{cases} \end{aligned}$$

Note. In the case of $i = j$: If $x_j > 1$ then our function becomes $c_j \lambda_j$. If $x_j \leq 1$ then our function becomes $c_j \lambda_j x_j$

Note. We note that if x_j the dependence on the chosen action disappears.

Amendment if using Instantly moving patroller If we are using the instantly moving patroller then we in fact get

For $s_j = B_j$

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

For $s_j = B_j + 1$

$$C_j(\boldsymbol{s}, \boldsymbol{v}, i) = \begin{cases} c_j \lambda_j \text{ for } i \neq j \\ 0 \text{ for } i = j \end{cases}$$

In our deterministic case we can further reduce the state space, as we choosing to visit later rather than earlier (as long as its not too late) allows us to possibly catch more (as we know when the attacks can start to finish). So we limit the state space with non-zero observed attackers to only have $s_j = B_j, B_j + 1$, as visiting at then gets any attacks caught when visiting at any $s_j < B_j$.

So in the deterministic case $\Omega = \{(B_j, v) \,|\, v = 0, 1, ..., b_j\} \cup \{(B_j + 1, 0)\}$.

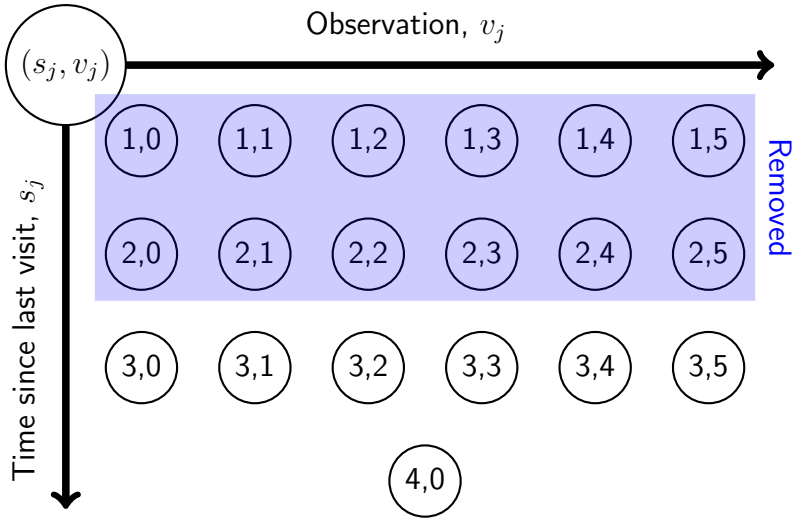

Figure 3.1: Deterministic state space diagram, with $b_j = 5$ and $B_j = 4$

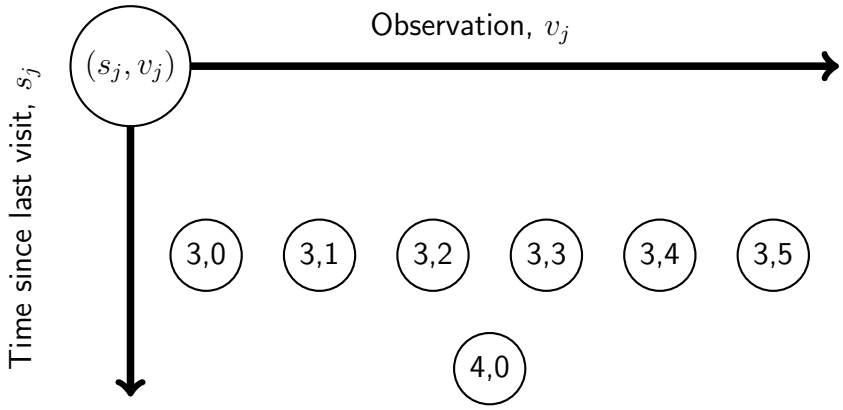

Figure 3.2: Final deterministic, with $b_j = 5$ and $B_j = 4$

3.1 Single node

We will now try to 'solve' the single node problem, when we have a deterministic attack time. We will drop all subscripts for the node.

We will now present a concrete argument as to why the decision is always to wait when in a state (s, v) with $s < B$. From this position consider the policy π_k which waits k time periods and then renews and follows the optimal policy, σ, with $k = 0, ..., B - s$.

Using such a policy will get us that

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

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

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

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

Where g is the average long-run costs of the system, coming from $V_n(x, v) = ng + \phi(x, v)$ for large n, and $\phi(x, v)$ is some initial bias for not starting in equilibrium from the initial state (x, v).

Now as the Dynamic Programming only has positive costs, it is impossible for $g < 0$. So we have that $g \geq 0$, so it is better to pick policy π_{k+1} over π_k, as this argument holds for all $k = 0, ..., B + 1 - s$ (and all v) it is best to wait till (B, v) before making a decision.

This formally shows the removal of the prior states in our single node problem.

Now we will skip to the state $(B + 1, 0)$ and suggest again a policy π_k which waits k time periods before renewing and then follows some optimal policy, σ.

Using such a policy will get us

$$V_n^{\pi_k}(x, v) = \omega + c\lambda k + E[V_{n-k-1}^{\sigma}(\theta)] \tag{5}$$

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

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

So hence if $g \geq c\lambda$ we will wait forever, as this has no dependence on k.

We will now argue that $g_{max} = c\lambda$, that is at worst the long-run average cost is $c\lambda$. This can be seen by the strategy π_{neg}, a strategy which never renews no matter what state we are in. I.e the patroller neglects the node. We will have that, for large n.

$$V_n^{\pi_{\text{neg}}}(s, v) = nc\lambda + \phi(s, v)$$

We can observe this by looking at the actual decay of the process, under a policy that never renews

$$\begin{aligned}
V_n^{\pi_{\text{neg}}}(s, v) =& V_{n-(B+1-s)}^{\pi_{\text{neg}}} + \phi(s, v) \\
&= (n - (B + 1 - x))c\lambda + \phi(s, v)
\end{aligned}$$

Where $\phi(x,v) = \underbrace{cv}_{\text{Observed finishing}} + \underbrace{c\lambda R}_{\text{arrivals that finish}}$ $(R = B - \lceil x \rceil)$ is the 'transfer'/bias of moving from the initial state to $(B+1,0)$.

Hence for any service cost, ω we can achieve a $g = c\lambda$. So we will always be in the case of $g \leq c\lambda$ and hence we will renew if we hit the state $(B+1,0)$ and hence for a state (B,v) we have two types of policy, π_0 a policy which renews now and then follows some optimal policy σ or policy, π_1 which waits one period until $(B+1,0)$ and then renews and follows some optimal policy σ.

So we will choice to renew now over waiting if

$$\begin{aligned}
&\lim_{n\to\infty} V_n^{\pi_1}(B,v) - V_n^{\pi_0}(B,v) \geq 0 \\
&\iff \lim_{n\to\infty} c\lambda R + vc + E[V_{n-1}^{\sigma}(B+1,0)] - (\omega + E[V_{n-1}^{\sigma}(\theta)]) \geq 0 \\
&\iff \lim_{n\to\infty} c\lambda R + vc + E[\omega + V_{n-2}^{\sigma}(\theta)] - \omega - E[V_{n-1}^{\sigma}(\theta)] \geq 0 \\
&\iff \lim_{n\to\infty} c(\lambda R + v) + E[V_{n-2}^{\sigma}(\theta) - V_{n-1}^{\sigma}(\theta)] \geq 0 \\
&\iff (\lambda R + v)c - g \geq 0 \\
&\iff g \leq c(\lambda R + v)
\end{aligned}$$

So we renew now if $g \leq c(\lambda R + v)$, as we are guaranteed that $g \leq c\lambda$ we are clearly in this region if $c(\lambda R + v) \leq c\lambda \iff v \leq \lambda(1-R)$. We also note that the bound is increasing in v so if we will renew in v we definitely renew in $v+1, v+2, ..., b$.

We notice that there is some critical value such that we renew for a collection of v's equal or higher and do not renew for lower v's. This critical value is $v_{\text{crit}} = \left\lceil \frac{g}{c} - \lambda R \right\rceil$ such that for $v \geq v_{\text{crit}}$ we renew immediately and for $v < v_{\text{crit}}$ we wait one time period and renew.

Note. We renew in one period as we are assumed to have $g \leq c\lambda$ otherwise we will just choice to neglect.

We have two special cases for g, that is when we always renew immediately or always wait then renew, these are $g \leq c\lambda R$ (i.e we renew for all v if we renew for $v = 0$) and $g \geq c(\lambda R + b)$ (i.e we always wait then renew for all v if we wait for $v = b$)

However between these regions, we have a divide of the nodes into a set we renew immediately and a set we wait then renew.

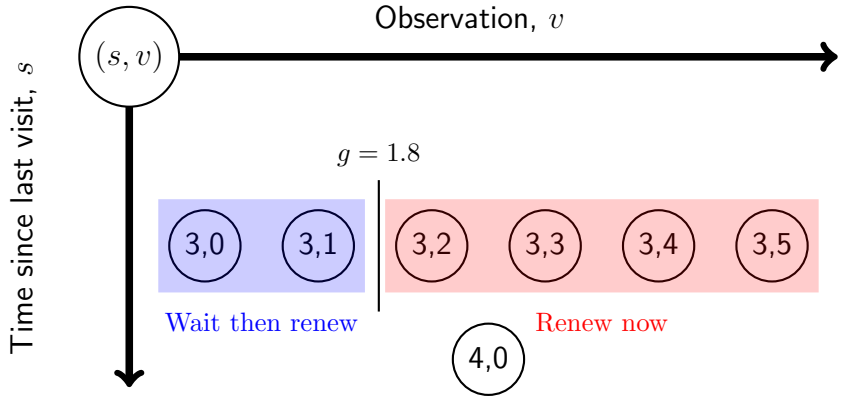

Figure 3.3: With $c = \lambda = 1$, $x = 4.4 \implies R = 0.6$ so $v_{\text{crit}} = 2$ when $1.6 < g < 2.6$ (e.g $g = 1.8$)

However we are assuming we are in the case of $g \leq c\lambda$ for so the example 3.3 we cannot have a situation as $g_{\max} = 1$. Really in this case we would just follow the neglecting strategy, π_{neg}.

We really need to split the region for g (which is bounded above by $c\lambda$) into different $v = v_{\text{crit}}$.

Say now we have that $b > \lambda(1-R)$ then we have some $v_{max} = \max\{v \in \{0,1,...,b\} \mid v \leq \lambda(1-R)\}$, some critical maximum (as at some point we will turn on the neglecting strategy just past it)

We do by having

- $v_{\text{crit}} = 0$ if $g \leq c\lambda R$

- $v_{\text{crit}} = k$ if $c\lambda R + c(k-1) < g \leq c\lambda R + kc$ for $k = 1, ..., v_{\max}$
- $v_{\text{crit}} = v_{\max} + 1$ if $c\lambda R + cv_{\max} < g \leq c\lambda$

Let us say that our $v_{\text{crit}} = k$ then we have that

$$\begin{aligned} g_k(\omega) &= \frac{\text{Expected cost per renewal}}{\text{Expected renewal length}} \\ &= \frac{\omega P(TPo(\lambda, b) \geq k) + (\omega + c\lambda R)P(TPo(\lambda, b) = 0) + ... + (\omega + c\lambda R + c(k-1))P(TPo(\lambda, b) = k-1)}{B \times P(TPo(\lambda, b) \geq k) + (B+1)P(TPo(\lambda, b) \leq k-1)} \\ &= \frac{\omega P(TPo(\lambda, b) \geq k) + (\omega + c\lambda R)P(TPo(\lambda, b) \leq k-1) + c\sum_{i=0}^{k-1} iP(TPo(\lambda, b) = i)}{B \times P(TPo(\lambda, b) \geq k) + (B+1)(1 - P(TPo(\lambda, b) \geq k))} \\ &= \frac{\omega + c\lambda R(1 - P(TPo(\lambda, b) \geq k)) + c\sum_{i=0}^{k-1} iP(TPo(\lambda, b) = i)}{B + 1 - P(TPo(\lambda, b) \geq k)} \end{aligned}$$

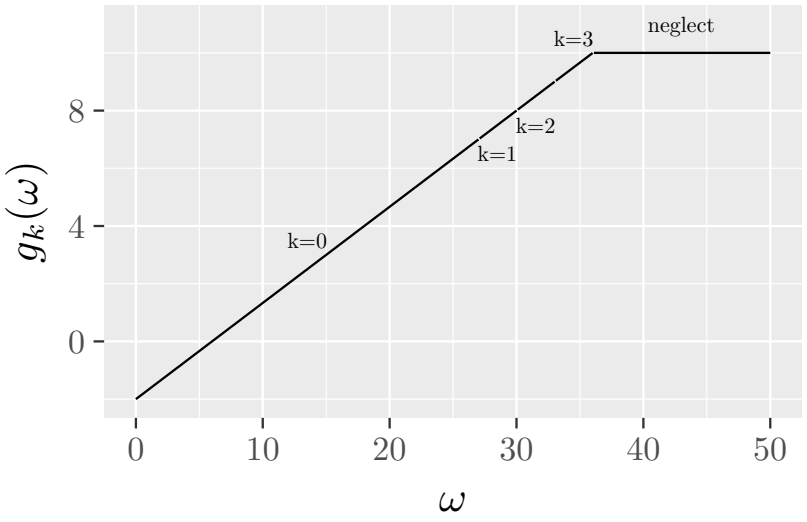

Figure 3.4: This shows the best long-run average cost (as a choice of k for each ω) Note. Figure needs checking against new formula (using B instead of floor(B)

Now we can translate our bounds on g to become bounds on ω

For $v_{\text{crit}} = 0$ we have $\delta(0) \equiv 0 \leq \omega \leq c\lambda R(B+1) \equiv \Delta(0)$.

For $v_{\text{crit}} = k$ we have $\delta(k) < \omega \leq \Delta(k)$

For $v_{\text{crit}} = v_{\max} + 1$ we have $\delta(v_{\max} + 1) < \omega \leq \widetilde{\Delta}$

Where

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

Now we really want consistent bounds for ω (i.e to have no over or under lap), so we would like $\delta(k) = \Delta(k-1)$, luckily by considering the formula's this is true.

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

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

$$= c(\lambda RB + (k-1)(B+1-P(TPo(\lambda,b) \geq k-1)) - \sum_{i=0}^{k-2} iP(TPo(\lambda,b)=i)) = \Delta(k-1)$$

We can now create an index based on our bounds for ω Then if $\omega \in [\Delta(k-1), \Delta(k)]$ ($\Delta(-1) = \delta(0) = 0$ by declaration) we should pick $V_{\text{crit}} = k$ and when $\omega \geq \widetilde{\Delta}$ we should pick to never visit the node.

This gives us a 'fair price' to select where to put the boundary of v_{crit}. To the left of this boundary we will wait and renew, to the left of the boundary we will renew immediately.

So now the question is given that we are in the state (B, v) will we renew now or not, well if we select a v_{crit} such that $v \geq v_{\text{crit}}$ then we will renew, otherwise we will elapse and renew (assuming $g \leq c\lambda$). Now we for $v \geq v_{\text{crit}}$ it means that the 'fair price' to renew is $\omega \leq \Delta(v)$ (that is we will select at least v_{crit} to be lower than or equal to v).

To conclude:

- If we are in the state (s, v) for $s < B, \forall v$ we will renew now if $\omega = 0$.
- If we are in the state (B, v) for $v < v_{\max}$ we will renew now if $\omega \leq \Delta(v)$.
- If we are in the state $(B, v_{\max})$ then we will renew now if $\omega \leq \widetilde{\Delta}$.
- If we are in the state (B, v) for $v > v_{\max}$ then we will renew now if $\omega \leq \widetilde{\Delta}$.
- If we are in the state $(B+1, 0)$ then we will renew now if $\omega \leq \widetilde{\Delta}$.

NOT SURE ABOUT THE LAST STATEMENT!!!!: seems to make sense coming through the prior state, but worried it does not work with the idea in this state we are paying $c\lambda$ every time unit.

3.2 Index Heuristic

To develop a heuristic based on indices we attach a suffix to the prior index which helps us to decide if it is worth visiting now or later.

Using the idea of the 'fair price' given by the bounds on ω for the single node we will define the index heuristic.

$$W_i(s_i, v_i) = \begin{cases} 0 \text{ If } s_i < B_i \\ \Delta_i(v_i) \text{ If } s_i = B_i, v_i < v_{i,\max} \\ \widetilde{\Delta_i} \text{ If } s_i = B_i, v_i \geq v_{i,\max} \\ \widetilde{\Delta_i} \text{ If } s_i = B_i + 1 \end{cases} \tag{6}$$

As the index is only non-zero in state (B_i, v) or $(B_i + 1, 0)$ and in the state (B_j, v) is dependent on the observed attackers v. We now decide that we will try to account for time, as the patroller doesn't really want to visit node j until it has been $s_j = B_j$ we will simply split the index up into even contributions for time step.

$$W_i(s_i, v_i) = \begin{cases} \frac{s_i \Delta_i(v_i)}{B_i} \text{ If } s \leq B_i, v_i < v_{i,\max} \\ \frac{s_i \widetilde{\Delta_i}}{B_i} \text{ If } s \leq B_i, v_i \geq v_{i,\max} \\ \widetilde{\Delta_i} \text{ If } s_i = B_i + 1 \end{cases} \tag{7}$$

We could suggest an alternative index that doesn't evenly split the index among the steps, and instead has a focus still on increasing towards the state of $s_i = B_i$, so the increments that they increase are increasing themselves.

$$W_i(s_i, v_i) = \begin{cases} \frac{\Delta_i(v_i)}{B+1-s_i} \text{ If } s \leq B, v_i < v_{i,\max} \\ \frac{\widetilde{\Delta}_i}{B+1-s_i} \text{ If } s \leq B_i, v_i \geq v_{i,\max} \\ \widetilde{\Delta}_i \text{ If } s_i = B_i + 1 \end{cases} \tag{8}$$

Note. Because the set of Δ's are monotone, this index is monotone in both s_i and v_i. We also note that $W_i(B_i, v) = W_i(B_i + 1, 0)$ for any $v_i \geq v_{\max}$, this value is our maximum pricing.

To implement the heuristic we initially assume that the patroller has neglected the region for a long period of time, that will be to set $s_i = B + 1$ (and hence assume all our state is $(\boldsymbol{s}, \boldsymbol{v}) = (B_1 + 1, ..., B_n + 1, 0, ..., 0)$, calculate the index for all the nodes and pick the node with the largest index (that is we will start at the node i which has the largest $\widetilde{\Delta}_i$). We then precede from here by calculating the indices of all adjacent nodes and moving to the node with the highest index, we repeat this process.

As the index is monotone and increasing in time, we will have that a nodes index grows (up to a cap of $\widetilde{\Delta}_i$) with time. However upon visiting it is not necessarily true that its it returns to the lowest value (as it may go to a state with a high v_i causing it to be higher).

As we only have control over when we visit, we note that the value $W_i(s_i, v_i)$ is the value at which the patroller is indifferent from visiting at s_i or $s_i + 1$ time units.

3.3 Lower bound

Recall $C^{\text{OPT}} \geq C^{\text{TR}} \geq C(\omega)$. We will aim to compute the tighest bound by getting $C_{max}(\omega) = \max_{\omega \geq 0} C(\omega)$.

For a given ω we will define

$$M_i(\omega, v_i) = \begin{cases} \infty \text{ If } \omega > W_i(\lfloor B_i \rfloor + 2, 0) \\ \lfloor B_i \rfloor + 2 \text{ If } \omega \leq W_i(\lfloor B_i \rfloor + 2, 0), \omega < W_i(\lfloor B_i \rfloor + 1, v_i) \\ \lfloor B_i \rfloor + 1 \text{ If } \omega \leq W_i(\lfloor B_i \rfloor + 2, 0), \omega \geq W_i(\lfloor B_i \rfloor + 1, v_i) \end{cases} \tag{9}$$

So that $M_i(\omega, v_i)$ represents the optimal interval to return to the node i when the last visit observed v_i attackers.

$$K_i(\omega) = \begin{cases} v_{i,\max} \text{ If } \omega \geq W_i(\lfloor B_i \rfloor + 2, 0) \\ \min\{k \in \{0, 1, ..., v_{i,\max}\} \,|\, W_i(\lfloor B_i \rfloor + 1, k) > \omega\} \end{cases} \tag{10}$$

So that $K_i(\omega)$ represents the optimal place to place the boundary for v_{crit}.

Note. $v_{\max} = \max\{v \in \{0, 1, ..., b + 1\} \,|\, v \leq \lambda_i R_i\}$, defined as before with a suffix.

We can let $C(\omega) = \sum_{i=1}^{n} C_i(\omega) - \omega$ where $C_i(\omega)$ is the optimal long run cost rate for node i when charged ω to visit. That is $C_i(\omega) = \min \frac{\text{Expected cost per renewal}}{\text{Expected renewal length}}$ and as we know that $M_i(\omega, v_i)$ is the optimal choice to renewal when we just observed v_i, as we know $v_i \sim TPo(\lambda_i, b_i)$ i.e we know the proportion of the time and how long we will wait.

We will first note that if $\omega > W_i(\lfloor B_i \rfloor + 2, 0)$ then we will get that $C_i(\omega) = c_i \lambda_i$. So when $\omega \leq W_i(\lfloor B_i \rfloor + 2, 0)$ we will have our old formula (dependent on placing the v_{crit} which will now be done optimally). So we will get that $C_i(\omega) = g_{K_i(\omega)}(\omega)$.

$$C_i(\omega) = \begin{cases} c_i\lambda_i \text{ If } \omega > W_i(\lfloor B_i \rfloor + 2, 0) \\ g_{K_i(\omega)}(\omega) \text{ Otherwise} \end{cases} \tag{11}$$

$$= \begin{cases} c_i\lambda_i \text{ If } \omega > W_i(\lfloor B_i \rfloor + 2, 0) \\ \frac{\omega - c_i\lambda_i(1-R_i)P(TPo(\lambda_i,b_i) \geq K_i(\omega)) + \sum_{j=0}^{K_i(\omega)-1} jP(TPo(\lambda_i,b_i)=j)}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i,b_i) \geq K_i(\omega))} \text{ Otherwise} \end{cases} \tag{12}$$

We now note some things

- $C_i(\omega)$ must be non-decreasing in ω, as the node can always do better with a smaller service charge.
- $C_i(\omega)$ is piecewise linear with turning points occurring at $\omega = W_i(\lfloor B_i \rfloor + 1, k)$ (and $\omega = W_i(\lfloor B_i \rfloor + 2, 0)$)
- $C_i(\omega)$ is concave because its functions are made from the lower envelope of straight line segments who's gradient is only increasing as turning points are passed ?????????????

Hence we have that $C(\omega) = \sum_{i=1}^{n} C_i(\omega) - \omega$ is also piecewise linear and concave, so computing $C_{max}(\omega)$ is straightforward, as the solution is either a line segment (non-unique ω maximizes) or a turning point (unique ω maximizes)

Suppose it is on a line segement, then $K_i(\omega)$ is constant and hence $C_i'(\omega) = \frac{1}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i,b_i) \geq K_i(\omega))}$ so $C(\omega) = \sum_{i=1}^{n} \frac{1}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i$ 1 and hence we need to find the optimal ω^* by solving

$$\sum_{i=1}^{n} \frac{1}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i, b_i) \geq K_i(\omega^*))} = 1$$

.

When the solution is unique, then we need to find the $\omega^* = W_i(k)$ for some i, k such that

- $\omega < \omega^* \iff \sum_{i=1}^{n} \frac{1}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i,b_i) \geq K_i(\omega^*))} > 1$
- $\omega > \omega^* \iff \sum_{i=1}^{n} \frac{1}{\lfloor B_i \rfloor + 2 - P(TPo(\lambda_i,b_i) \geq K_i(\omega^*))} < 1$

Then we have that $C^{\text{TR}} = C(\omega^*)$.

3.4 A special case of graph with two nodes

Consider now a simple graph, with two nodes. Using $\omega = \omega^*$ (to maximize the lagrangian) For node one given we have just observed, v_1 we will want a policy which returns in $M_1(\omega^*, v_1)$, call this type of policy π_1 and for node two given that we have just observed, v_2 we will want a policy which returns in $M_2(\omega^*, v_2)$.

Some randomization between these two patterns denoted $\alpha \otimes \pi_1 + (1-\alpha) \otimes \pi_2$ will also achieve $C_{max}(\omega)$ and has the property that the visit rate is 1 on average. If we can show that the index heuristic (IH) produces the same pattern as $\alpha \otimes \pi_1 + (1-\alpha) \otimes \pi_2$ then the IH is optimal.

Lemma 3.1. *For $n = 2$ the index heuristic is optimal and $C^{IH} = C^{OPT} = C_{max}(\omega)$.*

We first note that this should be obviously as we can simply move between two nodes at will and visit one when the index suggests to, i.e when we will 'save' the most.

Proof. WLOG suppose $W_1(1,0) \geq W_2(1,0)$ (meaning that $W_1(1, v_1) \geq W_2(1,0) \, \forall v_1 \in \{0, 1, ..., b\}$). □

4 Bernoulli Attack time

Consider the case where $X = \begin{cases} x_1 \text{ with probability } p \\ x_2 \text{ with probability } 1-p \end{cases}$ then we apply the same logic to attempt to get a decision dependent on the visiting cost. We will assume without loss of generality that $x_2 > x_1$, then $B = x_2$. We will get some reduction of the state space as before, but it will not be as drastic, by applying the same logic there may be a gap between some states we will never choose to visit. We will formally show this reduction in the appendix **??**

We will limit the state space with non-zero observed attackers to have either $s_j = \lfloor x_1 \rfloor + 1$ or $s_j = \lfloor x_2 \rfloor + 1$, due to the first one catching all the attacks caught for any $s_j < \lfloor x_1 \rfloor + 1$ and the second one catching all attacks caught for any $\lfloor x_1 \rfloor < s_j < \lfloor x_2 \rfloor + 1$.

So in the Bernoulli case $\Omega\{(\lfloor x_1 \rfloor + 1, o) \,|\, o = 0, 1, ..., b\} \cup \{(\lfloor x_1 \rfloor + 1, o) \,|\, o = 0, 1, ..., b\} \cup \{(\lfloor x_1 \rfloor + 2, o)\}$.

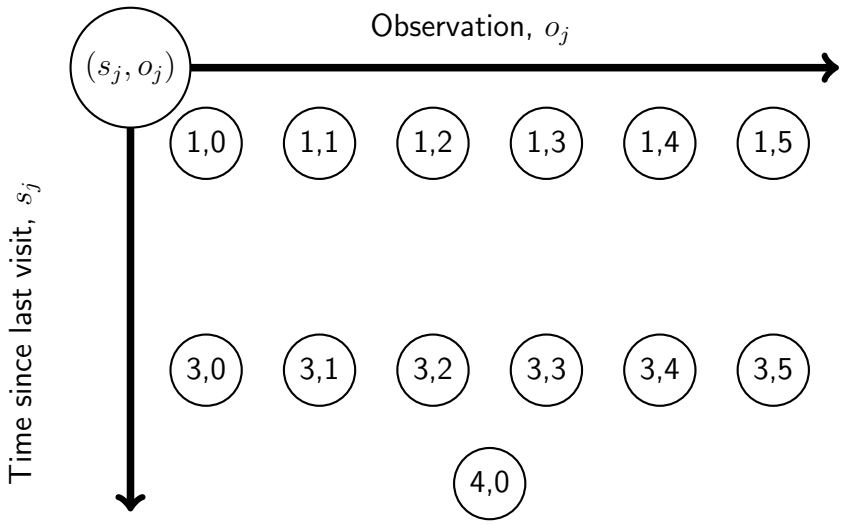

Figure 4.1: Bernoulli with $b_j = 5$, $x_{1,j} = 3.1$ and $x_{2,j} = 1.1$

We will be assuming that $\lfloor x_1 \rfloor \neq \lfloor x_2 \rfloor$ as otherwise it follows that only one s_j survives and we fall into the deterministic category (bar some changes to the values) see Apendix B.

Suppose we are in the state $(\lfloor x_1 \rfloor + 1, o)$ for some $o = 1, 2, ..., b$ then our decision is either to

- Visit now
- Wait till we are in state $(\lfloor x_2 \rfloor + 1, o)$ and then visit
- Wait till we are in state $(\lfloor x_2 \rfloor + 2, o)$ and then visit
- Wait till we are in state $(\lfloor x_2 \rfloor + 2, o)$ and wait

Now we write down the long-run average costs of following such a strategy

- Visit now:

$$\frac{c\lambda \int_0^{\lfloor x_1 \rfloor} P(X \leq t)dt + \omega}{\lfloor x_1 \rfloor + 1} = \frac{\omega}{\lfloor x_1 \rfloor + 1} \tag{13}$$

- Wait till state $(\lfloor x_2 \rfloor + 1, o)$ and then visit:

$$\begin{aligned} &\frac{c\lambda \int_0^{\lfloor x_2 \rfloor} P(X \leq t)dt + coP(X \leq \lfloor x_2 \rfloor) + \omega}{\lfloor x_2 \rfloor + 1} \\ &= \frac{c\lambda(\lfloor x_2 \rfloor - x_1)p + cop + \omega}{\lfloor x_2 \rfloor + 1} \end{aligned} \tag{14}$$

- Wait till state $(\lfloor x_2 \rfloor + 2, 0)$ and then visit:

$$\begin{aligned}&\frac{c\lambda \int_0^{\lfloor x_2 \rfloor + 1} P(X \leq t)dt + coP(X \leq \lfloor x_2 \rfloor + 1) + \omega}{\lfloor x_2 + 2 \rfloor} \\ &= \frac{c\lambda((x_2 - x_1)p + (\lfloor x_2 \rfloor + 1 - x_2)) + co + \omega}{\lfloor x_2 \rfloor + 2}\end{aligned} \tag{15}$$

- In state $(\lfloor x_2 \rfloor + 2, 0)$ waiting $k \geq 1$ then visiting:

$$\begin{aligned}&\frac{c\lambda \int_0^{\lfloor x_2 \rfloor + 1 + k} P(X \leq t)dt + coP(X \leq \lfloor x_2 \rfloor + 1 + k) + \omega}{\lfloor x_2 + 2 + k \rfloor} \\ &= \frac{c\lambda((x_2 - x_1)p + (\lfloor x_2 \rfloor + 1 + k - x_2)) + co + \omega}{\lfloor x_2 \rfloor + 2 + k}\end{aligned} \tag{16}$$

Very similar to before we will look at when certain costs are better. Starting with Visit now beating all others if $\omega < \frac{cp(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_2 \rfloor - x_1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor}$ and $\omega < \frac{c(\lfloor x_1 \rfloor + 1)(\lambda(x_2 - x_1)p + (\lfloor x_2 \rfloor - x_2 + 1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1}$ and $\omega < \frac{c(\lfloor x_1 \rfloor + 1)(\lambda(x_2 - x_1)p + (\lfloor x_2 \rfloor - x_2 + 1 + k) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1 + k}$ for all $k \geq 1$. The first inequality guarantees the other two so we get

Visit now if:

$$\omega < \frac{cp(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_2 \rfloor - x_1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor} \tag{17}$$

We can similarly find the visit in state $(\lfloor x_2 \rfloor + 1, o)$ by requiring that $\omega > \frac{cp(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_2 \rfloor - x_1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor}$ and $\omega < c(\lambda(p((x_2 - x_1)(\lfloor x_2 \rfloor + 1) - (\lfloor x_2 \rfloor - x_1)(\lfloor x_2 \rfloor + 2)) + (\lfloor x_2 \rfloor - x_2 + 1)(\lfloor x_2 \rfloor + 1) + o((1-p)(\lfloor x_2 \rfloor + 1) - p))$ and $\omega < \frac{c}{k+1}(\lambda(p((x_2 - x_1)(\lfloor x_2 \rfloor + 1) - (\lfloor x_2 \rfloor - x_1)(\lfloor x_2 \rfloor + 2 + k)) + (\lfloor x_2 \rfloor - x_2 + k + 1)(\lfloor x_2 \rfloor + 1) + o((1-p)(\lfloor x_2 \rfloor + 1) - p(k+1)))$. Note the second inequality implies the third one, so

Visit in state $(\lfloor x_2 \rfloor + 1, o)$ if:

$$\begin{aligned}&\frac{cp(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_2 \rfloor - x_1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor} < \omega < \\ &c(\lambda(p((x_2 - x_1)(\lfloor x_2 \rfloor + 1) - (\lfloor x_2 \rfloor - x_1)(\lfloor x_2 \rfloor + 2)) + (\lfloor x_2 \rfloor - x_2 + 1)(\lfloor x_2 \rfloor + 1) + o((1-p)(\lfloor x_2 \rfloor + 1) - p)) \\ &= c(\lambda(1-p)((\lfloor x_2 \rfloor - x_2)(\lfloor x_2 \rfloor + 1) + \lfloor x_2 \rfloor) + 1 + px_1 + o((1-p)(\lfloor x_2 \rfloor + 1) - p))\end{aligned} \tag{18}$$

Again we could possible not have this region if the left hand side overlaps the right (but for now we will ignore it).

Again we can similarly find the visit in state $(\lfloor x_2 \rfloor + 2, 0)$ by requiring that $\omega > \frac{cp(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_2 \rfloor - x_1) + o)}{\lfloor x_2 \rfloor - \lfloor x_1 \rfloor}$ and $\omega > c(\lambda(p((x_2 - x_1)(\lfloor x_2 \rfloor + 1) - (\lfloor x_2 \rfloor - x_1)(\lfloor x_2 \rfloor + 2)) + (\lfloor x_2 \rfloor - x_2 + 1)(\lfloor x_2 \rfloor + 1) + o((1-p)(\lfloor x_2 \rfloor + 1) - p))$ and $\omega < c(\lambda(1 + E[X]) - o)$. Now the second inequality implies the first one (???? double check this ????), so

Visit in state $(\lfloor x_2 \rfloor + 2, 0)$ immediately if:

$$\begin{aligned}&c(\lambda(1-p)((\lfloor x_2 \rfloor - x_2)(\lfloor x_2 \rfloor + 1) + \lfloor x_2 \rfloor) + 1 + px_1 + o((1-p)(\lfloor x_2 \rfloor + 1) - p)) \\ &< \omega < c(\lambda(1 + E[X]) - o)\end{aligned} \tag{19}$$

Again there is a possibility of overlap (but for now we will ignore it)

Finally we will never visit if:

$$\omega > c(\lambda(1 + E[X]) - o) \tag{20}$$

Example 4.1. Suppose $x_1 = 0.1$,$x_2 = 2.1$ with $p = 0.2$ and $b = 5$, then our state space is reduced to $\Omega = \{(1, o) \,|o = 0, 1, ..., 5\} \cup \{(1, o) \,|o = 0, 1, ..., 5\} \cup \{(4, 0)\}$ and we make the decision at state $(1, o)$. We start by calculating the various critical regions, for which we will use $\lambda = 20$ and $o = 1$.

First critical value(Equation 12): $\frac{c(0+1)(20 \times (2-0.1)+1)}{2-0} = \frac{39c}{2} = 19.5c$

Second critical value(Equation 13): $c(20 \times (1-0.2)((2-2.1)(2+1)+1) + 1 + 0.2 \times 0.1 + 1 \times ((1-0.2)(2+1) - 0.2)) = 14.42c$

Third critical value(Equation 14): $c(20 \times (1 + 0.2 \times 0.1 + 0.8 \times 2.1) - 1) = 53c$.

So in this case the decision will be based on the cost, ω:

- Visit immediately if $\omega \leq 19.5c$
- Wait till $(4, 0)$ (three time periods) and visit immediately if $19.5c < \omega < 53c$
- Never visit if $\omega \geq 53c$

4.1 Correction to approach for bernoulli

We will first have a concrete argument as to why we can renew all states, (s, v) with $s < \lfloor x_1 \rfloor + 1$. From this position consider the policy which waits k time periods and then renews and follows the optimally policy, σ, with $k = 0, ..., \lfloor x_1 \rfloor + 1 - s$.

Using such a policy will get us that

$$V_n^{\pi_k}(s, v) = \omega + E[V_{n-k-1}^{\sigma}(\theta)] \tag{21}$$

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

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

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

Where g is the average long-run costs of the system, coming from $V_n = ng + \phi(s, v)$ for large n, where $\phi(s, v)$ is some initial bias for not starting in the equilibrium but from the initial state (s, v).

Now as the Dynamic Programming only have positive costs, it is impossible for $g < 0$. So we have $g \geq 0$, so it is better to pick policy π_{k+1} over π_k , as the argument holds for all $k = 0, ..., \lfloor x_1 \rfloor + 1 - s$ (and all v) it is best to wait till $(\lfloor x_1 \rfloor + 1, v)$ before making a decision, i.e if we are too renew we might as well renew at $(\lfloor x_1 \rfloor + 1, v)$ (though we might not renew here).

This formally shows the removal of the initial section of states (i.e those at the top).

Now we will skip to the state $(\lfloor x_2 \rfloor + 2, 0)$ and again suggest a policy, π_k which waits k time periods before renewing and then follows some optimal policy, σ.

Using such as policy will get us

$$\begin{aligned}
V_n^{\pi_k}(s, v) &= \omega + c\lambda pk + c\lambda(1 - p)k + E[V_{n-k-1}^{\sigma}(\theta)] \\
&= \omega + c\lambda k + E[V_{n-k-1}^{\sigma}(\theta)]
\end{aligned}$$

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

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

So hence if $g \geq c\lambda$ we will wait forever, as this has no dependence on k.

We will now argue that $g_{\max} = c\lambda$, that is the worst long-run average cost is $c\lambda$. This can be be seen by the strategy, π_{neg}, a strategy which never renews no matter what state we are in. I.e the patroller neglects the node. We will have that, for large n

$$V_n^{\pi_{\text{neg}}}(s, v) = nc\lambda + \phi(s, v)$$

We can observe this by looking at the actual decay of the process under a policy that never renews

$$V_n^{\pi_{\text{neg}}}(s,v) = \begin{cases} nc\lambda \text{ If } s \geq \lfloor x_2 \rfloor + 2 \\ V_{n-(\lfloor x_2 \rfloor + 2 - s)} + \phi(s,v) \text{ If } s \leq \lfloor x_2 \rfloor + 1 \end{cases}$$
$$= \begin{cases} nc\lambda \text{ If } s \geq \lfloor x_2 \rfloor + 2 \\ (n - (\lfloor x_2 \rfloor + 2 - s))c\lambda + \phi(s,v) \text{ If } s \leq \lfloor x_2 \rfloor + 1 \end{cases}$$

where

$$\phi(s,v) = \begin{cases} \underbrace{vc}_{\text{Observed finishing}} + \underbrace{c\lambda p(\lfloor x_2 \rfloor - x_1 + 1)}_{x_1 \text{ type attackers finishing}} + \underbrace{c\lambda(1-p)(\lfloor x_1 \rfloor - x_1 + 1)}_{x_2 \text{ type attackers finishing}} \text{ If } s \leq \lfloor x_1 \rfloor + 1 \\ \underbrace{vc(1-p)}_{\text{Observed } x_2 \text{ types finishing}} + \underbrace{c\lambda p(\lfloor x_2 \rfloor - x_1 + 1 - s)}_{x_1 \text{ type attackers finishing}} + \underbrace{c\lambda(1-p)(\lfloor x_1 \rfloor - x_1 + 1)}_{x_2 \text{ type attackers finishing}} \text{ If } \lfloor x_1 \rfloor + 2 \leq s \leq \lfloor x_2 \rfloor + 1 \\ 0 \text{ If } s \geq \lfloor x_2 \rfloor + 2 \end{cases}$$
$$= \begin{cases} vc + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) + c\lambda p(x_2 - x_1) \text{ If } s \leq \lfloor x_1 \rfloor + 1 \\ vcp + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) + c\lambda p(x_2 - x_1 - s) \text{ If } \lfloor x_1 \rfloor + 2 \leq s \leq \lfloor x_2 \rfloor + 1 \\ 0 \text{ If } s \geq \lfloor x_2 \rfloor + 2 \end{cases}$$

is the 'transfer'/bias from the initial state to $(\lfloor x_2 \rfloor + 2, 0)$.

Hence for any service cost, ω we can achieve a $g = c\lambda$. So we will always be in the case of $g \leq c\lambda$ and hence will renew if we hit the state $(\lfloor x_2 \rfloor + 2, 0)$.

Now in this cases of $g \leq c\lambda$ (where we will not just follow neglection) we need to make a decision about what to do at states (s,v) for $\lfloor x_1 \rfloor + 1 \leq s \leq \lfloor x_2 \rfloor + 1$, $0 \leq v \leq b$.

The real question again is how long we wait, before renewal. Let us suppose we are in one of these states (s,v) then we notice that we will have slightly different costs depending which region we are in and how long we wait. We again use π_k be the policy to wait k ($k = 0, 1, ..., \lfloor x_2 \rfloor - \lfloor x_2 \rfloor + 1$) time units before renewing and then following some optimal policy, σ.

If we are in $(\lfloor x_1 \rfloor + 1, v)$ then we will get

$$V_n^{\pi_k}(\lfloor x_1 \rfloor + 1, v) = \begin{cases} \omega + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = 0 \\ \omega + vcp + c\lambda p(\lfloor x_1 \rfloor - x_1 + 1 + (k-1)) + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } 1 \leq k \leq \lfloor x_2 \rfloor - \lfloor x_1 \rfloor \\ \omega + vc + c\lambda p(\lfloor x_1 \rfloor - x_1 + 1 + \lfloor x_2 \rfloor - \lfloor x_1 \rfloor) + c\lambda(1-p)(\lfloor x_2 \rfloor - x_2 + 1) + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = \lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1 \end{cases}$$
$$= \begin{cases} \omega + E[V_{n-1}^{\sigma}(\theta)] \text{ If } k = 0 \\ \omega + vcp + c\lambda p(\lfloor x_1 \rfloor - x_1 + 1 + (k-1)) E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } 1 \leq k \leq \lfloor x_2 \rfloor - \lfloor x_1 \rfloor \\ \omega + vc + c\lambda p(x_2 - x_1) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) \text{ If } k = \lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1 \end{cases}$$

If we are in (s,v) for $1 \leq s \leq \lfloor x_2 \rfloor$

$$V_n^{\pi_k}(\lfloor x_1 \rfloor + 1, v) = \begin{cases} \omega + c\lambda pk + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } 0 \leq k \leq \lfloor x_2 \rfloor - \lfloor x_1 \rfloor - s \\ \omega + vc(1-p) + c\lambda p(\lfloor x_2 \rfloor - \lfloor x_1 \rfloor - s + 1) + c\lambda(1-p)(\lfloor x_2 \rfloor - x_2 + 1) + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = \lfloor x_2 \rfloor - \lfloor x_1 \rfloor + 1 - \end{cases}$$
$$= \begin{cases} \omega + c\lambda pk E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } 0 \leq k \leq \lfloor x_2 \rfloor - \lfloor x_1 \rfloor - s \\ \omega + vc(1-p) + c\lambda p(x_2 - \lfloor x_1 \rfloor - s) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) \text{ If } k = \lfloor x_2 \rfloor - \lfloor x_1 \rfloor - s + 1 \end{cases}$$

If we are in $(\lfloor x_2 \rfloor + 1, v)$

$$V_n^{\pi_k}(\lfloor x_2 \rfloor + 1, v) = \begin{cases} \omega + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = 0 \\ \omega + vc(1-p) + c\lambda p + c\lambda(1-p)(\lfloor x_2 \rfloor - x_2 + 1) + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = 1 \end{cases}$$
$$= \begin{cases} \omega + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = 0 \\ \omega + vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) + E[V_{n-k-1}^{\sigma}(\theta)] \text{ If } k = 1 \end{cases}$$

We now seek to compare policies for states, we we shall start with the state, $(\lfloor x_2 \rfloor + 1, v)$ then we will pick policy π_0 over π_1 if

$$\begin{aligned} &\lim_{n\to\infty} V_n^{\pi_1}(\lfloor x_2 \rfloor + 1, v) - v_n^{\pi_0}(\lfloor x_2 \rfloor + 1, v) \geq 0 \\ &\iff \lim_{n\to\infty} \omega + vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) + E[V_{n-2}^{\sigma}(\theta)] - (\omega + E[V_{n-1}^{\sigma}(\theta)]) \\ &\iff \lim_{n\to\infty} E[V_{n-2}^{\sigma}(\theta) - V_{n-1}^{\sigma}(\theta)] + vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) \\ &\iff g \leq vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor)) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1) \end{aligned}$$

So we will renew immediately in $(\lfloor x_2 \rfloor + 2, 0)$ from here if $g \leq vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor)) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1)$. We will use $R_2 = x_2 - \lfloor x_2 \rfloor$ $(0 \leq R_2 < 1)$.

So to have the possibility to wait and renew we will need that $vc(1-p) + c\lambda p R_2 + c\lambda(1-R_2) \leq c\lambda \iff c(1-p)(v - R_2\lambda) \leq 0 \iff v - R_2\lambda \leq 0 \iff v \leq R_2\lambda$.

Note. We note that this sort of result is analogous to the deterministic case.

now we note that we can again create a $v_{\text{crit}}(\lfloor x_2 \rfloor + 1)$, from which we will renew immediately for all $v \geq v_{\text{crit}}(\lfloor x_2 \rfloor + 1)$ and for $v < v_{\text{crit}}(\lfloor x_2 \rfloor + 1)$ we wait.

We rearrange $g \geq vc(1-p) + c\lambda p(x_2 - \lfloor x_2 \rfloor)) + c\lambda(\lfloor x_2 \rfloor - x_2 + 1)$ to get $v_{\text{crit}}(\lfloor x_2 \rfloor + 1) = \left\lceil \frac{g}{c(1-p)} - \frac{\lambda}{(1-p)} - \lambda R_2 \right\rceil$.

We again know that, $g_{\max} = c\lambda$ so we get the maximum $v_{\text{crit}} = \left\lceil \frac{c\lambda}{c(1-p)} - \frac{\lambda}{(1-p)} - \lambda R_2 \right\rceil = \lceil \lambda R_2 \rceil$

So when $R_2 = 0$ then we immediately renew and will never wait in the state $(\lfloor x_2 \rfloor + 1, v)$ for any v.

5 Ideas for future

- Extend X to be a binomial distribution
- Extend X to be any discrete distribution
- Extend X to be a normal distribution by approximation to binomial
- Test Deterministic case against strategic attacker (non-random) on extended star graph

Appendices

A Observations are always zero

On a single node we are limited to the state space of $\Omega = \{(1,0), ..., (\lfloor B \rfloor + 2, 0)\}$, then on this state space we can implement a policy which returns every k time units, this will gives an average long run cost of

$$f(k) = \frac{c\lambda \int_0^{k-1} P(X \leq t)dt + \omega}{k} \tag{22}$$

So to find out when the patroller would be indifferent from choosing to return every k or every $k+1$, solve $f(k+1)-f(k)=0$ giving

$$\frac{1}{k(k+1)}(c\lambda(k\int_{k-1}^{k}P(X\leq t)dt-\int_{0}^{k-1}P(X\leq t)dt)-\omega)=0$$

Prompting an index of

$$W(k)=c\lambda(k\int_{k-1}^{k}P(X\leq t)dt-\int_{0}^{k-1}P(X\leq t)dt)$$

We note that $W(0)=0$ and for $k\geq B+1$ $W(k)=c\lambda(k-int_{0}^{k+1}P(X\leq t)dt=c\lambda(1+\int_{0}^{k-1}P(X>t)dt)=c\lambda(1+E[X])$. We will now show that:

- $W(k)$ is non-decreasing
- The optimal policy when $\omega\in[W(k-1),W(k)]$ is to visit every k time units
- If $w\geq c\lambda(1+E[X])$ then it is optimal to never visit

Proof. First

$$\begin{aligned}W(k+1)-W(k)&=c\lambda((k+1)\int_{k}^{k+1}P(X\leq t)dt-\int_{0}^{k}P(X\leq t)dt\\&\quad-(k\int_{k-1}^{k}P(X\leq t)dt-\int_{0}^{k-1}P(X\leq t)dt))\\&=c\lambda((k+1)\int_{k}^{k+1}P(X\leq t)dt-k\int_{k-1}^{k}P(X\leq t)dt-\int_{k}^{k-1}P(X\leq t)dt)\\&=c\lambda(k+1)(\int_{k}^{k+1}P(X\leq t)dt-\int_{k-1}^{k}P(X\leq t)dt)\geq 0\end{aligned}$$

As $P(X\leq t)$ is non-decreasing.

Second if $\omega\in[W(k-1),W(k)]$ then we will show that $f(m)$ is non-increasing for $m\leq k$ and non-decreasing for $m\geq k$.

For $m\leq k$

$$\begin{aligned}f(m)-f(m-1)&=\frac{1}{m(m-1)}(c\lambda(m-1)\int_{0}^{m-1}P(X\leq t)dt-c\lambda m\int_{0}^{m-2}P(X\leq t)dt-\omega)\\&=\frac{1}{m(m-1)}(W(m-1)-\omega)\leq\frac{1}{m(m-1)}(W(m-1)-W(k-1))\leq 0\end{aligned}$$

Similarly for $m\geq k$

$$f(m+1)-f(m)=\frac{1}{m(m+1)}(W(m)-\omega)\geq\frac{1}{m(m+1)}(W(m)-W(k))\geq 0$$

Hence choosing to visit every k time units is optimal.

Third and finally our upper limit of $c\lambda(1+E[X])$ (Which is $W(k)$ for $k\geq B+1$) means we are indifferent from picking k and $k+1$ for $k\geq B+1$ so we will never visit. I.e $f(k+1)\leq f(k)\iff w\geq W(k)$ so not optimal if $f(k+1)\geq f(k)\,\forall k\iff w\geq\sup_{k=1,2,...}W(k)=\lim_{k\to\infty}W(k)=c\lambda(1+E[X])$ □

B Bernoulli Attack time with equal floors

If we have $\lfloor x_1\rfloor=\lfloor x_2\rfloor$ then $\Omega=\{(\lfloor x_1\rfloor+1,o)\,|\,o=0,1,...,b\}\cup\{(\lfloor x_1\rfloor+2,o)\}$

Suppose we are in the state $(\lfloor x_1\rfloor+1,o)$ for some $o=1,2,...,b$ then our decision process is the same as the deterministic case

- Visit now
- Wait till we are in state $(\lfloor x_1 \rfloor + 2, 0)$ and wait for k time units before visiting, $k \geq 0$

We again write down long-run average costs of following such a strategy

- Visit now:

$$\frac{c\lambda \int_0^{\lfloor x_1 \rfloor} P(X \leq t)dt + \omega}{\lfloor x_1 \rfloor + 1} = \frac{\omega}{\lfloor x_1 \rfloor + 1} \tag{23}$$

- Wait till state $(\lfloor x_1 \rfloor + 2, 0)$ then wait $k \geq 0$ time units before visiting:

$$\begin{aligned} &\frac{c\lambda \int_0^{\lfloor x_1 \rfloor + 1 + k} P(X \leq t)dt + coP(X \leq \lfloor x_1 \rfloor + 1 + k) + \omega}{\lfloor x_1 \rfloor + 2 + k} \\ &= \frac{c\lambda(p(x_2 - x_1) + (\lfloor x_1 \rfloor + 1 + k - x_2)) + co + \omega}{\lfloor x_1 \rfloor + 2 + k} \end{aligned} \tag{24}$$

Now to visit now we will only do so if $\omega < c(\lfloor x_1 \rfloor + 1)(\lambda(p\frac{x_2 - x_1}{k+1} + (\frac{\lfloor x_1 \rfloor - x_2}{k+1} + 1)) + \frac{o}{k+1})$ for all $k \geq 0$, We need to know if $\lambda(\lfloor x_1 \rfloor - px_1 + px_2 - x_2) + o \begin{cases} > 0 \\ < 0 \end{cases}$. In the first case we get to visit if:

$$\omega < c(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_1 \rfloor - px_1 + px_2 - x_2 + 1) + o) \tag{25}$$

In the second case we get to visit if:

$$\omega < c\lambda(\lfloor x_1 \rfloor + 1) \tag{26}$$

If we do no visit initially then we will visit once transitioned and waited $k + 1$ over k if $\omega > c(\lambda(1 + E[X]) - o)$.

For in between values (dependent on which case) we will get that we will transition and visit immediately. I.e if First case

$$c(\lfloor x_1 \rfloor + 1)(\lambda(\lfloor x_1 \rfloor - px_1 + px_2 - x_2 + 1) + o) < \omega < c(\lambda(1 + E[X]) - o) \tag{27}$$

Second case

$$c\lambda(\lfloor x_1 \rfloor + 1) < \omega < c(\lambda(1 + E[X]) - o) \tag{28}$$

Again these regions may or may not exist dependent on their conditions.