A Graph Patrol Problem with Locally-Observable Random Attackers

Thomas Lowbridge and David Hodge

University Of Nottingham,UK

June 15, 2018

Outline

- Introduction to game
  - Game setup
  - Markov Decision Process(MDP) formulation
- Problem relaxation
  - Multi-node relaxation
  - Total-rate relaxation
  - Lagrangain relaxation
- Single node solution
  - Developing an optimal policy
  - Developing a threshold policy
  - Fair costs
- Indices and heuristics

Game setup

A Patrolling game with random attackers and local-observations, $G = G(Q, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ is made of 5 major components.

- A Graph, $Q = (N, E)$, made of nodes, N ($|N| = n$), and a set of edges, E and an adjacency matrix, A.
- A vector of attack time distributions, $\boldsymbol{X} = (X_1, ..., X_n)$.
- A vector of observable capacities, $\boldsymbol{b} = (b_1, ..., b_n)$
- A vector of poisson arrival rates, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)$.
- A vector of costs, $\boldsymbol{c} = (c_1, ..., c_n)$

Game setup

A Patrolling game with random attackers and local-observations, $G = G(Q, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ is made of 5 major components.

- A Graph, $Q = (N, E)$, made of nodes, N ($|N| = n$), and a set of edges, E and an adjacency matrix, A.
- A vector of attack time distributions, $\boldsymbol{X} = (X_1, ..., X_n)$.
- A vector of observable capacities, $\boldsymbol{b} = (b_1, ..., b_n)$
- A vector of poisson arrival rates, $\boldsymbol{\lambda} = (\lambda_1, ..., \lambda_n)$.
- A vector of costs, $\boldsymbol{c} = (c_1, ..., c_n)$

The game is played over an infinite time horizon, $\mathcal{T} = 0, 1, ...$
A patroller's policy in the game is a walk (with waiting) on the graph,

$$W : \mathcal{T} \to N.$$

With the patroller moving instantaneously between nodes and attackers who arrive when the patroller is present waiting till the next time period to attack(who are observed as suspicious by the patroller).

Example of Game

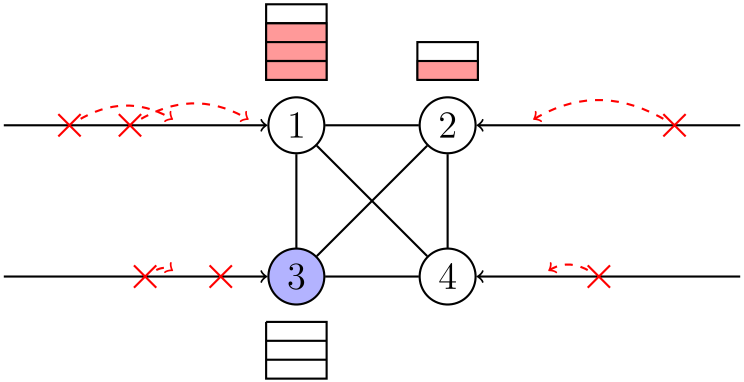

Figure 1: Example of $G = (K_4, \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{\lambda}, \boldsymbol{c})$ with patroller currently at node 3

Markov Decision Process(MDP) formulation

This game is a MDP with states $(\boldsymbol{s}, \boldsymbol{v})$, where s_i is the time since node i was last chosen to be visited and v_i is how many local-observations where observed when node i was last visited. With state space $\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) | s_i = 1, 2, ... \text{and}\, v_i = 0, 1, ..., b_i \,\forall i = 1, ..., n\}$.

The current node can be identified by $l(\boldsymbol{s}) = \text{argmin}_i\, \boldsymbol{s}$. The current node will have $s_i = 1$. The available actions from any state is $\mathcal{A}(\boldsymbol{s}) = \{j | A_{l(\boldsymbol{s}),j} = 1\}$.

A transition from a state,$(\boldsymbol{s}, \boldsymbol{v})$, with a chosen action, i, is $\phi(\boldsymbol{s}, \boldsymbol{v}, i) = (\widetilde{\boldsymbol{s}}, \widetilde{\boldsymbol{v}})$ where $\widetilde{s}$ has; $\widetilde{s}_i = 1$ and $\widetilde{s}_j = s_j + 1 \,\forall j \neq i$ and $\widetilde{v}$ has; $\widetilde{v}_i \sim TPo(\lambda_i)$ and $\widetilde{v}_j = v_i \,\forall j \neq i$.
Where $TPo(\lambda, b)$ is the Poisson distribution truncated at the value b. I.e

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

Note. $TPo(\lambda, \infty) = Po(\lambda)$.

Finiteness considerations

To avoid certain problems, we would like to limit ourselves to a finite state space

As in Lin et al. [2013], we will bound the attack times to create a finite state space, and define $B_j \equiv \min\{k | k \in \mathbb{Z}^+, P(X_j \leq k) = 1\}$. We will also bound our observable capacities to finite integers $b_i \in \mathbb{Z}_0^+$.This restricts our state space to a finite size $\Omega = \{(\boldsymbol{s}, \boldsymbol{v}) | s_i = 1, 2, ..., B_i + 1 \text{ and } v_i = 0, 1, ..., b_i \, \forall i = 1, ..., n\}$.

Now however our transistions are limited and become $\phi(\boldsymbol{s}, \boldsymbol{v}, i) = (\widetilde{\boldsymbol{s}}, \widetilde{\boldsymbol{v}})$ where $\widetilde{s}$ has; $\widetilde{s}_i = 1$ and $\widetilde{s}_j = \min\{s_j + 1, B_j + 1\} \forall j \neq i$ and $\widetilde{v}$ has; $\widetilde{v}_i \sim TPo(\lambda_i, b_i)$ and $\widetilde{v}_j = v_i \forall j \neq i$.

Example of state space

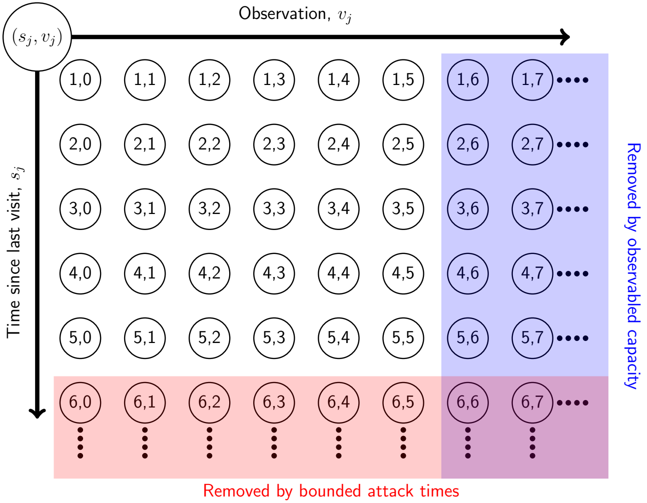

Figure 2: State space diagram, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Markov Decision Process(MDP) formulation

The cost at a node is zero if that node is chosen, otherwise it is the cost of arrivals finishing in the next time period, plus the cost of local-observations finishing in the next time period.

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

.

Due to the finiteness of the state space, we can just focus on stationary, deterministic policies, $\pi : \Omega \to \mathcal{A} \in \Pi$, instead of all walks, $\mathcal{W}$.

Deterministic attack times

We will now assume the attack times are deterministic, to make the cost function easier to handle.
When we use $P(X_j = x_j) = 1$ we get a cost function of.
For $s_j < B_j$

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

For $s_j = B_j$

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

Where $R_j = B_j - x_j$
For $s_j = B_j + 1$

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

Objective

The patroller wants to choose a policy such that the long-run average cost is minimized, that is find the minimum cost (and policy) defined by,

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

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

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

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

Problem Relaxation: Multiple Node

We now relax the problem to that of a patroller who can visit multiple nodes in the next time period.
We will call this Problem the multiple-node(MN) problem. We extend the class of polcies to

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

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. Note. $\Pi \subset \Pi^{\mathsf{MN}}$.

Total-rate constraint

Definition (Long-run visit rate)

Define the long-run rate of visits to node i, starting at $(\boldsymbol{s}_0, \boldsymbol{v}_0)$ under policy π by

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

We now impose the total-rate constraint, that is the long-run overall visit rate to all nodes is no greater than 1. and restrict the MN problem by this to get the total-rate(TR) problem and its class of policies

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

Again $\Pi \subset \Pi^{\mathsf{TR}}$.

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

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

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 (each costing ω).
We can show that, $C^{\mathsf{OPT}} \geq C^{\mathsf{TR}} \geq C(\omega)$.

Single node problem

We now want to find $C(\omega) = \min_{\pi \in \Pi^{\mathsf{MN}}} \sum_{i=1}^{n} (V_i(\pi) + \omega\mu_i(\pi)) - \omega$. This problem, as we are in the multi-node class, can be rewrote as

$$C(\omega) = \sum_{i=1}^{n} \min_{\pi \in \Pi^{\mathsf{MN}}} (V_i(\pi) + \omega\mu_i(\pi)) - \omega$$

That is for every node, i, try to minimize $V_i(\pi) + \omega\mu_i(\pi)$, where ω can be interpreted as the service charge for visiting the node.For now we drop the node subscript and just try to find the following,

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

We solve the single node problem, finding that $g \leq c\lambda$ by never visiting the node. We also find that depending on $g \leq c(\lambda R + v)$ we will renew or not when we are about to incur costs.

Developing a Threshold policy

This motivates a threshold policy as defined.

Definition (Threshold Policy)

Policy, $\pi_{\text{Th}}(v_{\text{crit}})$, is the policy which in states:

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

As $g \leq c(\lambda R + v)$ there is some maximum choice for this threshold, $v_{\text{max}} \equiv \max\{v \in \{0, 1, ..., b\} \,|\, v \leq \lambda(1 - R)\}$, so $v_{\text{crit}} \in \{0, 1, ..., v_{\text{max}} + 1\}$.
We get bounds on the long-run average cost

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

Example of a threshold policy

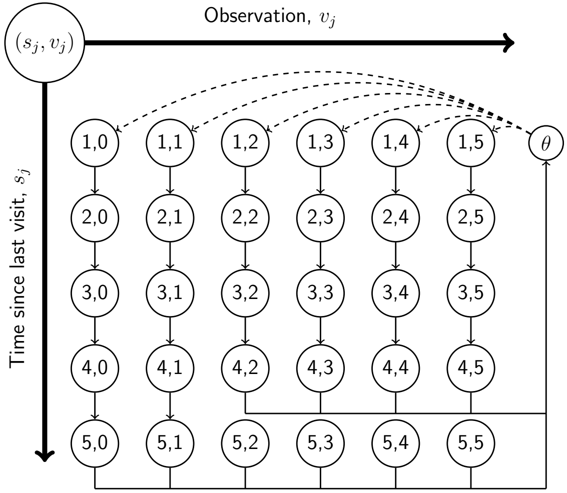

Figure 3: Threshold policy, $\pi_{\mathsf{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Converting bounds

By using such a strategy of v_{crit} threshold, we get a long-run average cost of

$$g^{\pi_{\text{Th}}}(\omega) = \frac{\text{Expected cost per renewal}}{\text{Expected renewal length}}$$

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

Converting the bounds on g to bounds on ω gives us the fair price

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

Where Δ,$\widetilde{\Delta}$ are appropriately defined by rearrangement. These now give us a fair cost to renew in (B, v) depending on v.

An index

We now reinsert the node subscript, i, we develop an index

Definition (Node Index)

We define the index for node i, when the system is in state, $(\boldsymbol{s}, \boldsymbol{v})$, to be

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

Example of node index

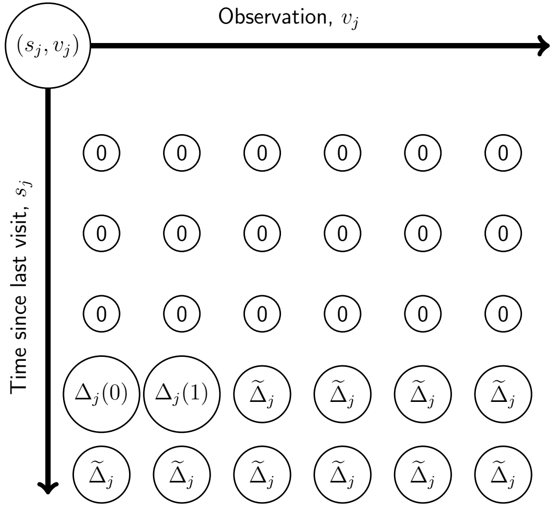

Figure 4: Index value for states

Heuristic types

Definition (Benefit Heuristic(BH))

The benefit heuristic(BH) has a patroller currently in state $(\boldsymbol{s}, \boldsymbol{v})$ look at all paths of length, l. They choose the path with the largest aggregate of indices collected along the path. The action is the first node in the path.

Definition (Penalty Heuristic(PH))

To develop a penalty heuristic(PH) based on the index, have a patroller currently in state $(\boldsymbol{s}, \boldsymbol{v})$ look at all paths of length, l. They choose the path with the smallest aggregate of indices not collected along the path. The action is the first node in the path.

Heuristic depth

If we are looking at all paths of length d, we might as well consider applying the heuristic to all paths of lengths, $1, ..., d$. We call this the heuristic depth.
We are left with d paths and therefore d suggestions for the first action to take. To decide on which action to take, we look at the average cost the path incurs and from the state at the end of the path we look at the average cost to decay to the neglected state $(\boldsymbol{B}, \boldsymbol{v})$.
We will call the depth heuristics, BH(d) and PH(d).

Numerical Experiments

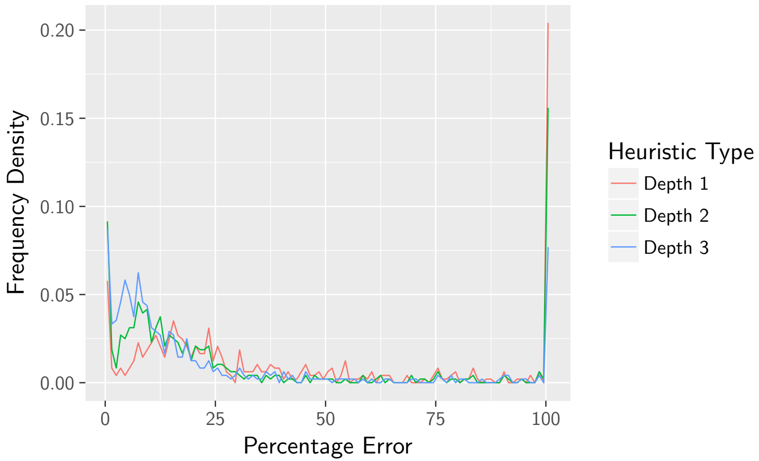

Figure 5: Percentage Error frequency for PH of depths $1, 2, 3$

Suggestion of node index fix

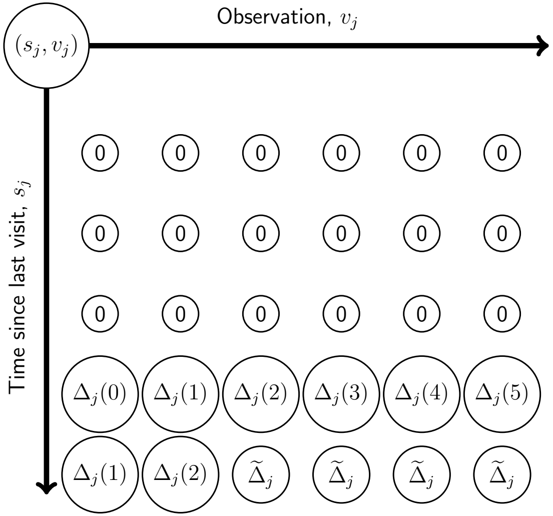
Observation, v_j
(s_j, v_j)
Time since last visit, s_j
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
$\Delta_j(0)$ $\Delta_j(1)$ $\Delta_j(2)$ $\Delta_j(3)$ $\Delta_j(4)$ $\Delta_j(5)$
$\Delta_j(1)$ $\Delta_j(2)$ $\widetilde{\Delta}_j$ $\widetilde{\Delta}_j$ $\widetilde{\Delta}_j$ $\widetilde{\Delta}_j$

Figure 6: New suggested index value for states

Comparing indices

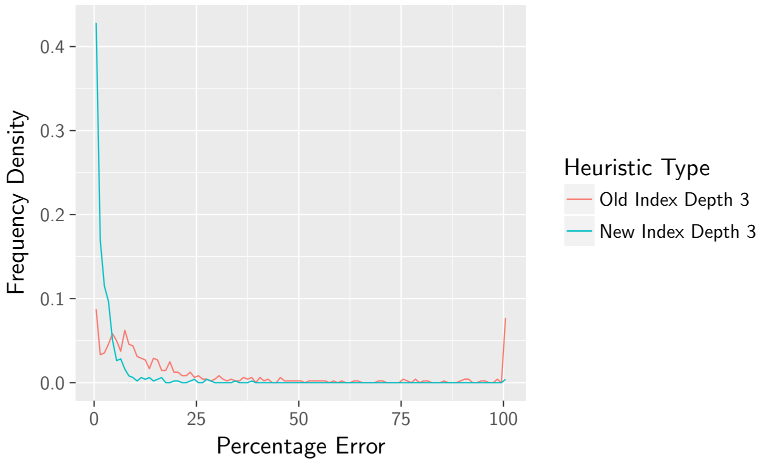

Figure 7: Percentage Error frequency for PH of depth 3 for old and new index

Future work

- Recode heuristic for each look-ahead window, l, to find the cycles and therefore average. This will make sure the heuristics are weakly improving in d.
- Look at how to pick a heuristic type dependent on graphical structure
- Improve theory to deal with generic distributions

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

Appendix: Developing the optimal policy

Developing an optimal policy

We aim to develop and index on our state space, that is a fair price for visiting the node for a given state. We shall first show there is no fair price, ω for visiting in any 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

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

where θ is the state upon a visit (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 long-run average cost and so we know $g \geq 0$ hence we will always wait instead of renewing.

Developing and optimal policy

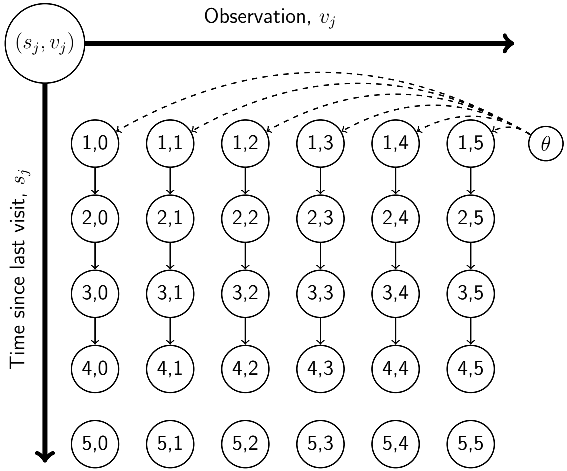

Figure 2.1: Threshold policy, $\pi_{\text{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Developing an optimal policy

Now we will skip to the state $(B+1, v)$ 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

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

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

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

So hence if $g \geq c\lambda$ we will wait forever, as this has no dependence on k. We will now argue that $g_{\text{max}} = c\lambda$, 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. As for large n we just pay $c\lambda$ every time step. Hence in these states if we renew we will renew immediately.

Developing an optimal policy

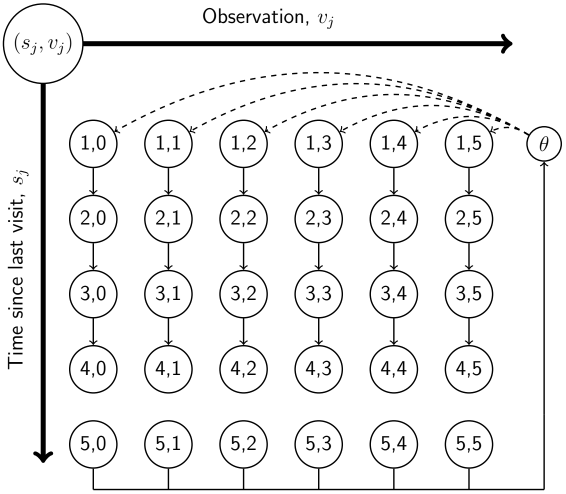

Figure 2.2: Threshold policy, $\pi_{\text{Th}}(2)$, with $b_j = 5$ and $B_j = 4$ (e.g. $X_j \leq 3.7$)

Developing an optimal policy

Our neglecting strategy shows that $g \leq c\lambda$ and hence if we are in state (B, v), we can decide to renew now or renew in one time period (at $(B+1, v)$) then follow the optimal, σ
we will choice to renew now over waiting if

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

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 $v+1, v+2, ..., b$.