Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Stochastics and Statistics

On the dynamic allocation of assets subject to failure

Stephen Ford[a,*], Michael P. Atkinson[b], Kevin Glazebrook[c], Peter Jacko[c]

[a] STOR-i Centre for Doctoral Training, Lancaster University, United Kingdom
[b] Operations Research Department, Naval Postgraduate School, United States
[c] Department of Management Science, Lancaster University, United Kingdom

ARTICLE INFO

Article history:
Received 18 June 2019
Accepted 10 December 2019
Available online 13 December 2019

Keywords:
Control
Dynamic programming
Heuristics
Queueing

ABSTRACT

Motivated by situations arising in surveillance, search and monitoring, in this paper we study dynamic allocation of assets which tend to fail, requiring replenishment before once again being available for operation on one of the available tasks. We cast the problem as a closed-system continuous-time Markov decision process with impulsive controls, maximising the long-term time-average sum of per-task reward rates. We then formulate an open-system continuous-time approximative model, whose Lagrangian relaxation yields a decomposition (innovatively extending the restless bandits approach), from which we derive the corresponding Whittle index. We propose two ways of adapting the Whittle index derived from the open-system model to the original closed-system model, a naïve one and a cleverly modified one. We carry out extensive numerical performance evaluation of the original closed-system model, which indicates that the cleverly modified Whittle index rule is nearly optimal, being within 1.6% (0.4%, 0.0%) of the optimal reward rate 75% (50%, 25%) of the time, and significantly superior to uniformly random allocation which is within 22.0% (16.2%, 10.7%) of the optimal reward rate. Our numerical results also suggest that the Whittle index must be cleverly modified when adapting it from the open-system, as the naïve Whittle index rule is not superior to a myopic greedy policy.

1. Introduction

The allocation of assets in an efficient way is an omnipresent problem. It is challenging because of the typical feature that the number of available assets is limited; such limited assets usually refer to a workforce or machines available to perform a particular task.

When assets are relatively simple machines connected to an energy supply, such as computers, it is reasonable to assume that they are available for continuous operation. Allocation of such assets to tasks has been extensively studied in the performance evaluation literature employing queueing theory techniques (see e.g. Harchol-Balter, 2013) and in the stochastic optimization literature employing dynamic programming techniques (see, e.g., Powell, 2007; Stidham & Weber, 1993).

However, considering a human workforce or cutting-edge machine technology, assets are rarely available for continuous operation. Such assets may be complex and hence failure-prone and time-consuming to repair, specialised and hence hard to quickly replace in case of an abandonment, or of limited endurance and hence regularly require recharging/refuelling. In this paper we focus on the allocation of such assets that tend to fail and must go through a replenishment process before they are once again available for operation.

1.1. Motivating examples

There is a broad range of situations in which assets are subject to failure.

Surveillance and patrolling: Consider a number of Unmanned Aerial Vehicles (UAVs) deployed to monitor a large area. The controller may want to keep surveillance at all times, so as not to allow gaps that could be exploited by an adversary. In this kind of situation, one could easily have tens to hundreds of UAVs, with time-scales stretching from minutes to days. The UAVs have a finite endurance and so must periodically return to base for refuelling and maintenance. For a review of the general UAV surveillance problem, see Nigam (2014); a similar situation arises in patrolling, see, e.g., Lin, Atkinson, and Glazebrook (2014).

* Corresponding author.
E-mail addresses: s.ford1@lancaster.ac.uk (S. Ford), mpatkins@nps.edu (M.P. Atkinson), k.glazebrook@lancaster.ac.uk (K. Glazebrook), p.jacko@lancaster.ac.uk (P. Jacko).

https://doi.org/10.1016/j.ejor.2019.12.018

Search and rescue: Similar difficulties occur in search and rescue operations, with the added complication that the assets may often be of different types such as foot teams, search dogs and helicopters. In this case, there may be only a small number of targets to be found, and finding them quickly could be of extreme importance. One such situation arises in a naval context, where UAVs can search for downed airmen in an ocean. See, for example (Cook, 2007; Waharte & Trigoni, 2010). Search assets cannot operate indefinitely; they need to occasionally rest (for humans or animals) or refuel (for vehicles).

Environment and wildlife monitoring: In environment and wildlife monitoring the controller needs to monitor an area for some time, either to get reliable readings or to ensure a sufficient sample of the flora or fauna in question. These are an example of why failure must inevitably be dealt with: it is simply impractical to survey the entire areas at once, and so some form of higher-level coordination is necessary. See, e.g., Casbeer, Beard, McLain, Li, and Mehra (2005) and Gonzalez et al. (2016).

Vehicle rental: The recent surge of sharing and short-term renting of environmentally-friendly vehicles (such as bicycles, drones, electric scooters, cars, etc.) in mobility-on-demand systems brings similar challenges. These vehicles are distributed to different locations where they wait (passively search) for customers. They frequently require repairs, cleaning and relocation. See, e.g., Boyacı, Zografos, and Geroliminis (2015) and Schuijbroek, Hampshire, and Van Hoeve (2017).

Queueing systems: More generically, consider a set of queues, where the controller assigns servers (assets) to a number of queues (tasks) to deal with arriving customers. The servers can only serve so many customers before needing a break, giving rise to failure and replenishment. Literature on queueing systems with server vacations and breakdowns is extensive (e.g. Doshi, 1986), but their control has only been addressed in specific cases; see for example Federgruen and So (1991) or Ke (2003).

Project management. If assets are employees or contractors, a manager needs to assign them to projects composed of tasks. People can only work for so long before needing a break, a rest or taking a leave. Such features are typically ignored in the project management literature (see Kerzner, 2017). There might be significantly more tasks than assets and decisions are made on a scale of weeks or months.

1.2. Problem description

In the rest of the paper we consider a generic problem in which a number of interchangeable assets need to be allocated to a number of different tasks. Any asset is allocatable to any task and several assets can be allocated concurrently to the same task.

An asset will fail after a random amount of time performing a task. The failure may manifest itself as the asset breaking down, draining its energy supply or being damaged. The failure requires repair, which occurs via recharging, refuelling, taking a break or replacement. The repair time is also random. During this repair time the asset cannot be allocated to any task. Once an asset is repaired, it is kept in reserve ready for allocation to any task, either immediately or later. Assets cannot be freely moved from one task to another, only allocated from the reserve to any of the tasks.

Every task yields a reward at a rate depending on the number of deployed assets, which is assumed non-decreasing and concave. The objective is to find a way of allocating assets to tasks that will maximise the long-term time-average sum of per-task reward rates.

It would be possible to formulate this problem without a reserve, and require that upon being repaired, an asset must be allocated to one of the k tasks. While this would yield a smaller state-space, for many reward functions, particularly those which are strongly concave, the optimal policy does keep a number of assets in the reserve. In Sections 3 and 4 we develop alternative policies that do not reserve assets. In Section 5 we perform a numerical comparison of the optimal policy versus these alternative policies and find that while in most scenarios the reserve option provides little benefit, in some cases it is vital.

1.3. Contributions and paper structure

In Section 2 we formulate this problem as a closed-system continuous-time Markov decision problem (MDP) with impulsive controls. To overcome the theoretical issues with impulsive controls, we reformulate it equivalently by replacing the impulsive controls with actions that do not immediately lead to system transitions, and further uniformise and discretise the problem. The resulting formulation allows for numerical solution by dynamic programming of small problem instances. It also allows for defining a greedy policy which myopically maximises the sum of reward rates.

Relevant to this formulation is research on optimisation of closed systems with multiple tasks. Many papers have used the framework of queueing networks (see, e.g., Stidham & Weber, 1993), but these allow or even require assets to move between tasks and only one asset is allowed to be active at a task (the other assets wait in a queue). In terms of optimisation, the focus is mainly on queue scheduling disciplines (which does not apply in our model) and on service rate control (which is somewhat related to our model as the number of allocated assets to a task affects both the failure rate and the reward).

For any reasonably-sized problem, standard dynamic programming techniques are computationally infeasible. In Section 3 we therefore develop an alternative formulation of the problem as an open-system approximation to the original closed-system problem. This open-system imagines that assets depart from the system when they fail and new assets arrive independently to the system, as if from the repair process. This model is related to the standard problem of routing/dispatching to parallel queues (see, e.g. Hyytiä, 2013), in which assets correspond to jobs and tasks to queues with servers, with the difference that each task is shared by all the allocated assets rather than following a queueing discipline. We innovatively extend the restless bandits approach of (Whittle, 1988) to derive its Lagrangian relaxation and decomposition, which leads to a parametric single-task problem that can be seen as a problem of admission control to a multi-server queueing system. Section 3.2 is then devoted to the derivation of the Whittle index. A similar model and solution approach appeared in Borkar and Pattathil (2017), with the difference that the reward from the task is constant in the number of assets deployed.

In Section 4 we describe two ways of adapting the Whittle index derived from the open-system model to the original closed-system model, a naïve one and a cleverly modified one. To the best of our knowledge these are both novel. In order to prepare ground for numerical evaluation of these index rules, we describe the uniformly random policy and the greedy policy, and explain how the optimal policy can be found when it is practical to do so.

Section 5 then describes comprehensive numerical experiments of performance evaluation of these policies in the original closed-system model, which indicates that the cleverly modified Whittle index rule is nearly optimal, being within 1.6% (0.4%, 0.0%) of the optimal reward rate 75% (50%, 25%) of the time, and significantly superior to uniformly random allocation which is within 22.0% (16.2%, 10.7%) of the optimal reward rate (see Table 1). Our numerical results also suggest that it is crucial for the Whittle index to be cleverly modified when adapting it from the open-system to the closed-system model, since the naïve Whittle index rule is not

Table 1
Mean and five-number summary given as the relative suboptimality gap (in percent).

Policy	Mean	Min.	5th Percentile	Lower Quartile	Median	Upper Quartile	Max.
Uniformly Random	17.1	52.1	34.1	22.0	16.2	10.7	0.0
Greedy	3.6	58.3	18.4	3.7	0.5	0.0	0.0
Naïve Whittle	4.6	60.6	14.5	6.3	2.3	0.0	0.0
Clever Whittle	1.3	26.6	5.4	1.6	0.4	0.0	0.0

superior to the greedy policy which is obtained by myopically optimising the original closed-system model.

Section 6 presents limitations and concludes with a discussion of possible extensions of this work. A single overly long proof is deferred to Appendix A, while pseudocode for the policy iteration algorithm to optimally solve the MDP is given in Appendix B.

2. The problem formulation

In this section we formalise the problem outlined in the previous section as a continuous-time Markov decision process (MDP) (see Puterman, 2014, chapter 11); that is, for the dynamics we focus on the Markovian case. There are N interchangeable assets to allocate to K different tasks.

2.1. Markov decision process model

A continuous-time MDP operating over time $0 \le t \le H$, where H is the time horizon, is defined by its states, actions, transition rates, and reward rates.

States: We denote the current number of assets at task $k = 1, \dots, K$ by x_k. For notational convenience, we write x_{K+1} for the number of assets under repair, and x_{K+2} for the number of assets currently repaired but not yet allocated – the number currently reserved. The reserve does in fact sometimes prove necessary, especially if the reward functions are very concave. Assets are discrete so that $x_k \in \mathbb{N}_0$ – the non-negative integers – for all $k = 1, \dots, K+2$, and the total number of assets is constant so that $\sum_{k=1}^{K+2} x_k = N$. We denote the current overall state by $\boldsymbol{x} = (x_1, x_2, \dots, x_{K+2})$, and the space of possible states by $\mathcal{X}$.

Actions: After it reaches the reserve task, any asset can then immediately be allocated from the reserve to any of the tasks, or be held back for later allocation. The actions $\boldsymbol{a}$ are the allocations of assets from reserves to tasks $k = 1, \dots, K$, which move the state instantaneously from $\boldsymbol{x}$ to $\boldsymbol{x} + \boldsymbol{a}$. We write $\mathcal{A}(\boldsymbol{x})$ for the space of currently feasible actions when in state $\boldsymbol{x}$; it is defined by the following four constraints:

(i) $a_k \ge 0,\ k = 1, \dots, K+1$; previously allocated assets or assets that are being repaired may not be reallocated;
(ii) $-x_{K+2} \le a_{K+2} \le 0$; assets from the reserve can be allocated up to the number currently there;
(iii) $\sum_{k=1}^{K+2} a_k = 0$; the total number of assets is constant;
(iv) $a_k \in \mathbb{Z}\ \forall k$; assets are discrete.

Note that we may take the action 0 even if we have assets in the reserve: we do not have to allocate all the immediately available assets.

We specify the actions to be chosen for each state by a policy u. A policy u is a map, possibly non-deterministic, from states $\boldsymbol{x} \in \mathcal{X}$ to actions $\boldsymbol{a} \in \mathcal{A}$. The family of admissible policies $\mathcal{U}$ includes all non-anticipating policies prescribing actions satisfying $\boldsymbol{a} \in \mathcal{A}(\boldsymbol{x})$. We will in particular restrict ourselves to deterministic stationary policies, which always take the same action when in the same state. This can be done without loss of generality for MDPs (Puterman, 2014).

Transition rates: Apart from those caused by our actions, there are two types of transition: due to a failure and due to a repair. Neither the failure transition nor the repair transition depends on the current action. Failures happen on a per-asset basis, depending only on the task the asset is allocated to; we denote the failure rate for an asset allocated to task k by μ_k, with the natural constraint that all $\mu_k > 0$. Any asset that fails immediately joins the repair process, i.e. a failure in task k causes a transition from $\boldsymbol{x}$ to $\boldsymbol{x} - \mathbf{e}_k + \mathbf{e}_{K+1}$. Here $\mathbf{e}_k$ is the k^{th} standard $(K+2)$-dimensional unit basis vector.

The repair process produces ready-to-allocate assets one at a time at some non-decreasing state-dependent rate $\lambda(x_{K+1})$. The function $\lambda(x)$ is a function defined on $\mathbb{N}_0$, satisfying the following two constraints: $\lambda(0) = 0$ so no assets means no repairs, and $\lambda(x)$ must be non-decreasing so more assets being repaired makes repairs more likely to happen. After an asset is repaired, it transitions to the reserve status: repairs, which can happen as long as $x_{K+1} \ge 1$, cause a transition from $\boldsymbol{x}$ to $\boldsymbol{x} - \mathbf{e}_{K+1} + \mathbf{e}_{K+2}$.

The assets' time to failure and the time until the next asset is repaired both follow an exponential distribution. The times-to-failure are independent for each asset and independent of anything else, such as the overall state of the system. The repair process is independent of anything else as well.

Reward rates: Since the actions move the state instantaneously, for our purposes it is sufficient to consider reward rates that are independent of actions. We write that in state $\boldsymbol{x}$, the controller gains reward at a rate $R(\boldsymbol{x})$. We impose that the rewards are gained independently from each task; that is, the reward-rate function is of the following form:

$$R(\boldsymbol{x}) = \sum_{k=1}^{K} g_k(x_k) \tag{1}$$

The functions g_k represent the reward obtained from allocating assets to task $k = 1, \dots, K$. We impose three conditions of the g_k, which we expect to hold in practice:

(i) $g_k(x)$ is non-decreasing; more assets allocated to a task produces no less reward;
(ii) $g_k(x)$ is concave in x; there are diminishing marginal rewards;
(iii) $g_k(0) = 0$; no assets allocated gives no reward.

Note that there is no reward gained for assets under repair or in reserve.

Objective: The MDP is characterised by the state process $\boldsymbol{X}(\cdot)$ and the action process $\boldsymbol{A}(\cdot)$, induced by the transition rates and by the chosen policy u. We consider the long-term time-average objective for the MDP over an infinite horizon. We define V as the optimal long-term time-average reward rate, so that:

$$V = \max_{u \in \mathcal{U}} \liminf_{H \to \infty} \frac{1}{H} \mathbb{E}^u \left[\int_0^H R(\boldsymbol{X}(t)) dt \right] \tag{2}$$

This expression is well defined because the state-space is finite and the reward rates are non-negative and bounded above, since there is a finite number of assets. In particular, for stationary policies we will have that $\lim \equiv \liminf$ in the equation above.

2.2. Solution by dynamic programming

The type of actions defined above are known as impulsive controls for continuous-time MDPs. The main difficulty with dealing with impulsive control models is that the process can take several different values at the same time moment, which makes the classical theory of MDPs inapplicable.

The structure of our MDP model allows for a reformulation in which impulsive actions are replaced by actions that do not immediately lead to system transitions, by making both the transition rates and reward rates dependent on the action, as in Dufour and Piunovskiy (2015). This reformulation results in a merging of the state transition caused by the impulsive action with the next state transition due to failure or repair. Rather than viewing the system as transitioning instantaneously from $\boldsymbol{x}$ to $\boldsymbol{x}+\boldsymbol{a}$, we frame the system as remaining in state $\boldsymbol{x}$ with rewards and transitions dictated by $\boldsymbol{x}+\boldsymbol{a}$.

In particular, a failure of an asset allocated to task k while $\boldsymbol{a}$ prevails occurs at rate $(x_k+a_k)\mu_k$ and causes a transition from $\boldsymbol{x}$ to $\boldsymbol{x}+\boldsymbol{a}-\mathbf{e}_k+\mathbf{e}_{K+1}$. Similarly while $\boldsymbol{a}$ prevails, a repair occurs at rate $\lambda(x_{K+1}+a_{K+1})$ and causes a transition from $\boldsymbol{x}$ to $\boldsymbol{x}+\boldsymbol{a}-\mathbf{e}_{K+1}+\mathbf{e}_{K+2}$. Finally, while $\boldsymbol{a}$ prevails in state $\boldsymbol{x}$ the controller gains reward at a rate $R(\boldsymbol{x}+\boldsymbol{a})$.

Note that as we have instant controls, applying two actions a_1 and then a_2 is equivalent to applying a_1+a_2.

We can uniformise the problem, transforming the reformulated continuous-time model into an equivalent discrete-time model. Define $B=\lambda(N)+N\max\mu_k$, as an upper bound on the transition rate out of all states. Consider a related Markov decision process that transitions out of every state with rate B. For this related process, with probability $(x_k+a_k)\mu_k/B$ an asset in task k fails and with probability $\lambda(x_{K+1}+a_{K+1})/B$ an asset is repaired. With the remaining probability $[B-\lambda(x_{K+1}+a_{K+1})-\sum_{k=1}^{K}(x_k+a_k)\mu_k]/B$ the dummy transition moves the system to state $\boldsymbol{x}+\boldsymbol{a}$. During uniformisation the dummy transition usually has the system return to its current state; because of our impulsive action reformulation described in the previous paragraph, the dummy transition here goes to $\boldsymbol{x}+\boldsymbol{a}$. For more details on uniformisation, see Puterman (2014, Section 11.5).

This uniformisation results in a discrete-time MDP. We can present the optimality equation for the discrete-time MDP with a long-term time-average objective, utilising standard MDP machinery; see Puterman (2014, Chapters 8 and 11) for more details. We define $\phi(\boldsymbol{x})$ as the relative value of a state $\boldsymbol{x}$. This relative value is also referred to as the bias (see Puterman, 2014, Chapter 8), and is a measure of how much the controller prefers starting the system in one state compared to some reference state.

We obtain the following dynamic programming optimality equation, which is satisfied by the optimal value V for our problem:

$$\begin{aligned} &V+\phi(\boldsymbol{x})\\ &=\max_{\boldsymbol{a}\in\mathcal{A}(\boldsymbol{x})}\frac{1}{B}\Bigg(R(\boldsymbol{x}+\boldsymbol{a})+\lambda(x_{K+1}+a_{K+1})\phi(\boldsymbol{x}+\boldsymbol{a}-\mathbf{e}_{K+1}+\mathbf{e}_{K+2})\\ &\quad+\sum_{k=1}^{K}(x_k+a_k)\mu_k\phi(\boldsymbol{x}+\boldsymbol{a}-\mathbf{e}_k+\mathbf{e}_{K+1})\\ &\quad+\left(B-\lambda(x_{K+1}+a_{K+1})-\sum_{k=1}^{K}(x_k+a_k)\mu_k\right)\phi(\boldsymbol{x}+\boldsymbol{a})\Bigg)\quad\forall\boldsymbol{x}\in\mathcal{X} \end{aligned}\tag{3}$$

This is a set of $|X|$ equations for $|X|+1$ unknowns - the $\phi(x)$ and V. We hence choose some reference state $\mathbf{x}_0$ (in practice $(0,0,\dots,N)$) and set $\phi(\mathbf{x}_0)=0$. We can then solve the set of equations to find the optimal policy, and the optimal value V.

For definiteness, we define $\phi(\boldsymbol{x})=0$ for all $\boldsymbol{x}$ such that $x_k<0$ for any $k=1,\dots,K+2$: for all ineligible x the terms disappear.

On the LHS of (3), we have the optimal value plus the relative value of the current state. On the RHS, we choose the control $\boldsymbol{a}$ to maximise the sum of four terms. The first term is the immediate expected reward received in state $\boldsymbol{x}$ while $\boldsymbol{a}$ prevails. Note that $R(\boldsymbol{x}+\boldsymbol{a})$ is a rate, and hence $R(\boldsymbol{x}+\boldsymbol{a})/B$ is the expected reward accumulated before the next transition.

The second term captures the expected future reward obtained when a repair transition occurs next. The third term represents the expected reward from all failure transitions. The fourth term is the expected reward from the dummy transition where the system moves to state $\boldsymbol{x}+\boldsymbol{a}$.

In theory, we can compute the optimal V from (3) using standard MDP solution techniques such as policy iteration, value iteration, or linear programming (see Puterman, 2014 chapter 8). Unfortunately, in practice such exact methods quickly become intractable even for relatively small problems.

The state space for our problem has size $|\mathcal{X}|=\binom{N+K+1}{K+1}$ because the controller distributes N identical assets among $K+2$ non-identical tasks. For instance, if $N=15$ and $K=10$, then $|\mathcal{X}|$ is more than 7.7 million. Given this, computing the optimal solution will only be feasible for small problems, as standard policy iteration requires us to form a matrix of size $|\mathcal{X}|+1$ by $|\mathcal{X}|+1$. We now turn to approximative methods, in particular one that will allow us to consider the tasks separately.

3. A Restless bandit approximation

Restless bandits Whittle (1988) provide a framework applicable to our problem. A generalisation of the well-known multi-armed bandit problem Gittins (1979), a restless bandit is a specific type of MDP.

In a restless bandit problem there are K bandits, which are independent Markov decision processes with binary action space: the controller can choose to activate the bandit, or not to. Each bandit has its own state x_k and generates rewards at a rate depending only on its own state. The control is quite simple: at each decision point, the controller can activate up to $M<K$ of the bandits; the controller gains reward according to both the state of all the bandits and the actions taken. Taking the active action is usually better in some way, either providing more immediate reward, or charging up the bandit.

The restlessness is as follows: even the bandits not activated still evolve stochastically, usually decaying in some way. This contrasts with a classic bandit problem of the Gittins type, where the controller can only activate one bandit at a time, and the non-activated bandits remain in the same states.

3.1. The approximating system

We now formulate an approximating system, which modifies the original problem in three respects:

(i) We remove the repair process, and have assets arriving for allocation to tasks at a constant rate Λ independently of the system's state: assets arrive according to a Poisson process with rate Λ.
(ii) Failing assets depart the system entirely, as opposed to going to a repair buffer.
(iii) We remove the reserve, so arriving assets are immediately allocated.

The diagrams in Figs. 1 and 2 illustrate the difference between the original system and the approximating system.

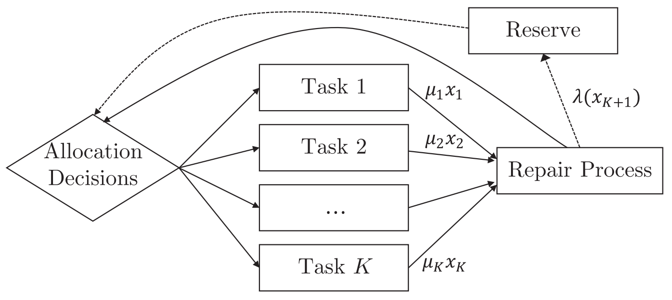


Fig. 1. A diagram of the problem. Assets fail, are repaired, and are then allocated to tasks, possibly via a sojourn in the reserve.

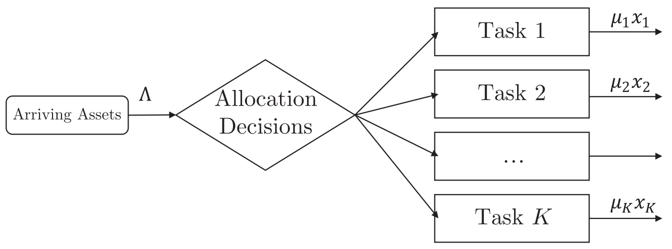


Fig. 2. A diagram of the restless bandit approximation. Assets arrive at a constant rate, and there is no reserve or repair process.

The approximating system is a continuous-time MDP with the elements as follows. We write x_k for the number of assets currently at task $k = 1, \ldots, K$. The number of assets is unbounded and the state space is $\mathbb{N}^K \geq 0$ – a vector of K non-negative integers. Each arriving asset is immediately allocated to one of K tasks. Assets at task k fail at rate μ_k independently of each other, and of everything else in the system. The aggregate asset failure rate in system state $\boldsymbol{x} = (x_1, x_2, \ldots, x_K)$ is given by $\sum_{k=1}^{K} \mu_k x_k$. Rewards are earned in state $\boldsymbol{x}$ at a rate $R(\boldsymbol{x}) = \sum_{k=1}^{K} g_k(x_k)$.

We now write $u : \mathbb{N}^K \to \{1, 2, \ldots, K\}$ for a stationary allocation policy; when the approximating system is in state $\boldsymbol{x}$, an arriving asset is assigned to task $u(\boldsymbol{x})$. We define $\mathcal{U}$ as the set of all such policies. We will write the optimum allocation problem for the approximating system as:

$$G^{opt} = \max_{u \in \mathcal{U}} \sum_{k=1}^{K} V_k^u \tag{4}$$

Here we write V_k^u for the long-term time-average reward rate earned by task k under allocation policy $u \in \mathcal{U}$, i.e.:

$$V_k^u := \lim_{H \to \infty} \frac{1}{H} \mathbb{E}^u \left[\int_0^H g_k(X_k(t)) dt \right]. \tag{5}$$

Even this simplified version of our original problem lies beyond the scope of analysis via conventional dynamic programming. However, we can make progress by using ideas which have their origins in the Whittle (1988) description of a class of restless bandit problems. We create a Lagrangian relaxation of the above allocation problem as follows: first extend the class of feasible policies from $\mathcal{U}$ to a class $\mathcal{U}' \supseteq \mathcal{U}$. Any $u \in \mathcal{U}'$ is given by a map $u : \mathbb{N}^K \to 2^{\{1,2,\ldots,K\}}$. Under $u \in \mathcal{U}'$ we allow each incoming asset to be allocated to all tasks in any subset (including the empty set) of the task set $\{1, 2, \ldots, K\}$, with the total number of assets assigned to each task in the nominated subset incrementing by one. Consequently, the arrival of one asset to the system can result in the allocation of up to K assets (one asset per task) for a policy $u \in \mathcal{U}'$.

We write $\Lambda_k(u)$ for the long-term time-average allocation rate of assets to task k under $u \in \mathcal{U}'$. We develop a relaxation of the above allocation problem in (4) as follows. Define:

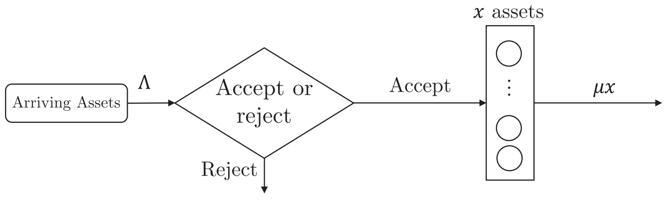


Fig. 3. A diagram of a single task of the restless bandit approximation.

$$G^{opt'} = \max_{u \in \mathcal{U}'(\Lambda)} \sum_{k=1}^{K} V_k^u \tag{6}$$

The maximum is taken over the policy set $\mathcal{U}'(\Lambda)$ given by:

$$\mathcal{U}'(\Lambda) = \left\{ u \in \mathcal{U}'; \sum_{k=1}^{K} \Lambda_k(u) \leq \Lambda \right\} \tag{7}$$

Problem (6) allows us to use the extended policy class $\mathcal{U}'$, but our total asset allocation rate is constrained by the arrival rate Λ of assets.

We now relax the problem in (4) further by dropping the constraint in (7) and instead incorporating it into the objective in a Lagrangian fashion to obtain

$$G^{opt}(W) = \max_{u \in \mathcal{U}'} \left\{ \sum_{k=1}^{K} (V_k^u - W\Lambda_k(u)) \right\} + W\Lambda \tag{8}$$

The multiplier W has an interpretation as a charge levied whenever an asset is allocated to a task. It is plain that

$$G^{opt}(W) \geq G^{opt'} \geq G^{opt}, \quad W \geq 0, \tag{9}$$

Further, it can be shown by a standard argument that:

$$\min_{W \geq 0} G^{opt}(W) = G^{opt'} \tag{10}$$

Further inspection of the problem in (8) makes it plain that due to independence of the tasks, the problem can be decomposed into K independent asset-allocation problems, given the Lagrange multiplier W.

See Fig. 3 for a diagram of the single-task problem. We will say that the single task problem is in a state x if there are x assets currently at the task. We write:

$$G^{opt}(W) = \sum_{k=1}^{K} G_k(W) + W\Lambda, \tag{11}$$

where:

$$G_k(W) = \max_{u_k \in \{0,1\}^{\mathbb{N}}} \left(V_k^{u_k} - W\Lambda_k(u_k) \right), 1 \leq k \leq K. \tag{12}$$

The $\{0,1\}^{\mathbb{N}}$ in (12) is a way to represent a policy u_k as an infinite dimensional binary vector where the policy assigns an incoming asset to task k in state i if index i of the binary vector is a 1, otherwise the policy declines the incoming asset if index i is 0.

The problem in (12) concerns task k only. It seeks an optimal allocation policy when assets arrive at task k according to a Poisson process of rate Λ and are either allocated to the task (action 1), or not (action 0) dependent upon the task's current state. Reward rates for task k are as in the original problem in (4), but now rewards are earned net of charges levied at a rate of W per allocation.

We can now focus attention on this parametric single-task problem, and until further notice shall drop the task identifier k from the notation. We thus write the single task problem as:

$$G(W) = \max_{u \in \{0,1\}^{\mathbb{N}}} (V^u - W\Lambda^u) \tag{13}$$

We write $u(W)$ for an optimal allocation policy for the subproblem in (13). We seek optimal policies of simple structure for the Lagrangian relaxation in (8) by looking for simplicity of structure in $u(W)$. The following notion (Whittle, 1988) is key:

Definition 1. The approximating system is indexable if there exist optimal policies $u(W)$ for all parametric single-task problems for which the activation sets:

$$A(W) = \{x | x \in \mathbb{N} \text{ and } u(W,x) = 1\} \tag{14}$$

are decreasing in W.

We prove in the following subsection that the approximating system is indexable and derive the corresponding Whittle index.

3.2. Indexability

Indexability is the natural requirement that, for each task, as the allocation charge W increases, the set of states in which it is optimal to allocate an incoming asset to the task decreases. Since the following result has a proof along the lines of a corresponding result in Glazebrook, Kirkbride, and Ouenniche (2009) for a different system, we shall provide a sketch only.

Proposition 1. *The approximating system is indexable.*

Proof. First, it is trivial to show that there exists optimal policies $u(W)$ of threshold type, namely whose corresponding allocation sets take the form:

$$A(W) = \{0, 1, \ldots, T(W) - 1\} \tag{15}$$

for some $T(W) \in \mathbb{N}$. Hence $T(W)$ is the minimal state for which the optimal policy dictates that an incoming asset should be rejected. Define $\pi_{T(W)}(x)$ as the stationary probability of being in state x, given a policy with threshold $T(W)$. For such a policy we have a long-term time-average rate of allocation of incoming assets equal to:

$$\Lambda(u(W)) = \Lambda\left\{1 - \pi_{T(W)}(T(W))\right\} \tag{16}$$

Here $\pi_{T(W)}(T(W))$ is the stationary probability that the task operating under policy $u(W)$ has its maximal number of assets present, namely $T(W)$. The dynamics of the task under this policy are a $M/M/\infty$ queue, truncated at $T(W)$, and so:

$$\pi_T(T) = \frac{\frac{1}{T!}\left(\frac{\Lambda}{\mu}\right)^T}{\sum_{x=0}^{T} \frac{1}{x!}\left(\frac{\Lambda}{\mu}\right)^x} \tag{17}$$

which is decreasing in T. We now remark that $G(W)$, being the upper envelope of a set of linear functions, must be convex in W. It then follows that $\Lambda(u(W))$ must be decreasing in W and hence from (16), $\pi_{T(W)}(T(W))$ must be increasing in W. It then follows that $T(W)$ and $A(W)$ must be decreasing in W. This establishes indexability, and so completes the proof. □

Remark 1. Note that the reward function g plays no role in the proof of the above proposition. Hence the approximating system is indexable whatever the functional form the rewards take. However, as we develop stronger notions of indexability, the increasing concave nature of g will come into play. We now proceed to define the indices which the indexability of the approximating system make possible.

Definition 2. An index function $W : \mathbb{N} \to \mathbb{R}^+$ for a task with associated optimal acceptance sets

$\{A(W) = \{0, 1, \ldots, T(W) - 1\}, W \in \mathbb{R}^+\}$ is defined by

$$W(x) = \sup\left\{W | W \in \mathbb{R}^+ \text{ and } x \in A(W)\right\} \tag{18}$$

Definition 3. The approximating system is strictly indexable if, for all tasks, the mapping $T : \mathbb{R}^+ \to \mathbb{N}$ which takes W to $T(W)$ is onto $\mathbb{N}$.

Remark 2. We already know from the above proposition that the mapping T is non-increasing.

Before proceeding further, we extend the notation in (17) by noting that the stationary distribution for the number of assets at a task operating under acceptance set $\{0, 1, \ldots, T - 1\}$ is given by

$$\pi_T(x) = \frac{\frac{1}{x!}\left(\frac{\Lambda}{\mu}\right)^x}{\sum_{y=0}^{T} \frac{1}{y!}\left(\frac{\Lambda}{\mu}\right)^y}, 0 \le x \le T \tag{19}$$

We now introduce the key quantities $w(x), x \in \mathbb{N}$, given by

$$w(x) = \frac{\sum_{y=0}^{x+1} g(y)\pi_{x+1}(y) - \sum_{y=0}^{x} g(y)\pi_x(y)}{\Lambda(\pi_x(x) - \pi_{x+1}(x+1))} \tag{20}$$

The quantity $w(x)$ has a natural interpretation as follows: imagine that the decision maker facing the single task problem in (13) is committed to accepting newly arriving assets provided there are no more than $x - 1$ of them currently present at the task. The question arises of the effect of expanding the acceptance set to include x as well.

The numerator in (20) is the increase in reward rate thus achieved while the denominator is the increase in acceptance rate. It is the case that when this marginal quantity is less than W (the charge per unit asset accepted) then the expansion of the acceptance rate proposed leads to a decrease in the overall reward rate in (13).

Should W be less than $w(x)$, then the reverse conclusion holds and the expansion proposed improves the overall reward rate. From these considerations it follows that, under suitable conditions on the reward function g, $w(x)$ coincides with the index $W(x)$ defined above. We now outline a proof of this key result.

Theorem 3. *If the reward function g is increasing and strictly concave, then the approximating system is strictly indexable with index given by $W(x) = w(x), x \in \mathbb{N}$.*

Proof. We introduce the quantities:

$$P_x = \sum_{y=0}^{x} \frac{\left(\frac{\Lambda}{\mu}\right)^y}{y!}, \qquad x \in \mathbb{N} \tag{21}$$

In Appendix A, we prove the identity:

$$w(x) = \frac{\sum_{y=0}^{x} P_y(g(y+1) - g(y))}{\mu \sum_{y=0}^{x} P_y} \tag{22}$$

from which it follows that $w(x)$ is strictly decreasing in x when g is increasing and strictly concave. We also note that $w(0) = \frac{g(1)}{\mu}$ and because g is concave, each asset considered separately can only give a reward of at most $\frac{g(1)}{\mu}$. We can therefore write:

$$0 \le \lim_{x\to\infty}(g(x+1) - g(x)) \le \lim_{x\to\infty} w(x) \le \frac{g(1)}{\mu} \tag{23}$$

We now write $G_x(W)$ for the reward rate achieved in (13) when the acceptance set is $\{0, 1, \ldots, x\}$. We have

$$G_x(W) = \sum_{y=0}^{x+1} g(y)\pi(y|x+1) - W\Lambda(1 - \pi(x+1|x+1)) \tag{24}$$

and we then obtain $W = w(x)$ as the unique $W-$solution to the equation:

$$G_x(W) = G_{x-1}(W), \quad x \in \mathbb{N} \tag{25}$$

with the natural interpretation for the case $x = 0$. Further, each $G_x(W)$ is linear in W with a gradient which is negative, and which decreases as x increases. From these facts it follows that for all choices of $x \in \mathbb{N}$:

$$G_x(W) < G_y(W), \quad y \le x - 1, W \ge w(x) \tag{26}$$

and

$$G_{x+1}(W) < G_y(W), \quad y \le x, W \le w(x+1) \tag{27}$$

From these inequalities and the fact that $w(x)$ is strictly decreasing in x, we can deduce that for any $x \in \mathbb{N}$:

$$G_x(W) > \max_{y \neq x} G_y(W), \quad w(x+1) \le W \le w(x), \tag{28}$$

Hence:

$$A(W) = \{0, 1, \dots, x\}, \quad w(x+1) \le W \le w(x) \tag{29}$$

with

$$A(w(x)) \in [\{0, 1, \dots, x-1\}, \{0, 1, \dots, x\}] \tag{30}$$

The corresponding rejection thresholds are given by:

$$T(W) = x+1, \quad w(x+1) \le W \le w(x) \tag{31}$$

and

$$T(w(x)) \in \{x, x+1\}. \tag{32}$$

It now follows from the above that the map T is onto $\mathbb{N}$, that the approximating system is strictly indexable, and that the index is given by:

$$W(x) = \sup\left\{W | W \in \mathbb{R}^+ \text{ and } x \in A(W)\right\} = w(x) \tag{33}$$

as required. This concludes the proof. □

To continue the discussion, we now restore the task identifier k. From the above discussion, under the assumption that all reward functions g_k are increasing and strictly concave, the approximating system is strictly indexable, and associated with each task k is an index function $W_k : \mathbb{N} \to \mathbb{R}^+$ given by a suitable form of the quantity in (20).

It is an immediate consequence of the above discussion that an optimal policy for the Lagrangian relaxation in (8) may be expressed as follows: when the approximating system is in state $\boldsymbol{x}$, an arriving asset should be allocated to those tasks for which $W_k(x_k) > W$. Denote this policy by *Index*(W). Plainly the asset allocation rate associated with *Index*(W) is decreasing in W. It is straightforward to show that the relaxed problem in (6) is solved by *Index*(W_Λ), where W_Λ is such that:

$$\sum_{k=1}^{K} \Lambda_k(Index(W_\Lambda)) = \Lambda \tag{34}$$

Please note that to achieve equality in (34) it may be necessary to introduce randomisation into the operation of *Index*(W_Λ) when a task-state pair for which $W_k(x_k) = W_\Lambda$ is encountered. In light of these optimal policies of index structure, we follow (Whittle, 1988) in proposing a solution for (4) in the form of an index rule for the approximating system as follows: when the approximating system is in state $\boldsymbol{x}$ an arriving asset should be allocated to any task $l \in \arg\max_k W_k(x_k)$.

4. Policies

To evaluate how well the Whittle index rule works on example problems, we will compare it against the optimal policy, when possible. However, as discussed earlier, it is only possible to compute the optimal value for smaller problems. Therefore, we also compare the Whittle index rule to two other policies which could easily be applied to larger problems. We now expound briefly on the policies, including how to actually apply the Whittle index rule to the original closed-system problem.

4.1. The Whittle index rule

Although we provide a full derivation and explanation of the Whittle index above, we touched only briefly on how to use this to make decisions for a given instance of the original closed-system problem. Recall from (20) that the formula for the Whittle index for task k which has x_k assets is (dropping the initial constant $1/\Lambda$):

$$w_k(x_k) = \frac{\sum_{x=0}^{x_k+1} \pi_{x_k+1}(x) g_k(x) - \sum_{x=0}^{x_k} \pi_{x_k}(x) g(x)}{\pi_{x_k}(x_k) - \pi_{x_k+1}(x_k+1)} \tag{35}$$

Defining $\rho_k \equiv \frac{\Lambda}{\mu_k}$ for the open-system, we rewrite the stationary distribution from (19)

$$\pi_{x_k}(x) = \frac{\rho_k^x / x!}{\sum_{y=0}^{x_k} \rho_k^y / y!} \tag{36}$$

The key input to compute the index $w_k(x_k)$ is the ratio ρ_k, which appears in (35) via the stationary distribution in (36). This ratio is not well defined in the original closed-system problem because Λ is not a parameter in that system. Consequently we need to define ρ_k more concretely for the original system. We examine two different Whittle index variants corresponding to different values for ρ_k; we denote one the *naïve* index and the other the *clever* index.

The naïve index sets $\rho_k = \frac{\lambda(x_{K+1}+1)}{\mu_k}$. This definition directly mimics the open-system definition of ρ_k by replacing the arrival rate Λ with the repair rate $\lambda(x_{K+1}+1)$, which is reasonable because assets arrive for allocation in the original system following a repair. We use $x_{K+1}+1$ instead of x_{K+1} because repaired assets move immediately to the reserve, and hence x_{K+1} could be zero when the allocation decision is made.

The clever index sets

$$\rho_k = \frac{\lambda(x_{K+1}+1)}{\lambda(x_{K+1}+1) + \mu_k}. \tag{37}$$

We choose this value of ρ_k for two reasons. First this value of ρ_k produces the optimal solution for the original closed-system in the case with $N = 1$ asset. As the rewards are unchanging, the optimal policy is to always allocate the asset to the same task. If the controller always allocates to task k, the asset is at that task for a fraction of the time $\lambda(1)/(\mu_k + \lambda(1))$. The controller gets a reward per unit time of

$$g_k(1) \frac{\lambda(1)}{\lambda(1) + \mu_k}, \tag{38}$$

and the optimal policy allocates to the task which makes the expression in (38) the largest. In this case, the Whittle index for task k for the state $x_k = 0$ can be found from Eqs. (35) and (36)

$$w_k(0) = \frac{\pi(1|1) g_k(1)}{1 - \pi(1|1)} = g_k(1) \frac{\frac{\rho_k}{1+\rho_k}}{1 - \frac{\rho_k}{1+\rho_k}} = g_k(1)\rho_k \tag{39}$$

Comparing (39) to (38), the Whittle index rule produces the optimal policy if we define ρ_k according to Eq. (37).

The second justification for the ρ_k defined in (37) is that the naïve index is obviously not capturing the system dynamics correctly when $\lambda(x_{K+1}+1) \gg \mu_k$. In this case, the naïve index makes assignments believing that future assets will be arriving for allocation rapidly and indefinitely; in reality the actual arrival rate in the closed-system is constrained by the finite number of assets in the system. The clever index addresses this issue by replacing the arrival rate Λ in the open-system not just with the repair rate as the naïve index does, but with a throttled version of the the repair rate:

$$\Lambda \leftarrow \lambda(x_{K+1}+1) \frac{\mu_k}{\lambda(x_{K+1}+1) + \mu_k}. \tag{40}$$

We then define $\rho_k = \frac{\Lambda}{\mu_k}$ as in the open-system, which produces the desired quantity in (37). We can view (40) as stating the arrival rate to the approximation system (Λ) is the repair rate $(\lambda(x_{K+1}+1))$ multiplied by a rough estimate of the fraction of time an asset is under repair, assuming an alternating renewal framework specifically, $\frac{\mu_k}{\lambda(x_{K+1}+1)+\mu_k}$.

Regardless of how ρ_k is chosen, the procedure for using the Whittle index rule is as follows. Suppose the system is in state $\boldsymbol{x}$: the Whittle index rule does not keep a reserve, so the controller must wait until a repair completes to make a decision. We then proceed as follows:

Step 1: For each task k, note the number x_k of assets at that task;
Step 2: Set ρ_k, using either the naïve or clever approach described above.
Step 3: Calculate $w_k(x_k)$ using Eq. (35), using ρ_k in the calculation of the stationary probabilities in (36).
Step 4: Allocate the asset to the task with the highest calculated Whittle index $w_k(x_k)$; in case of ties choose the task first in the ordering.

4.2. Policies for comparison

We shall compare the two Whittle index rule variants with three other policies: uniformly random allocation, greedy allocation, and where possible the optimal policy.

Uniformly random allocation is the simplest: whenever there are assets in the reserve, the controller allocates them randomly, with equal probability for each task. This policy is not intended to be a sensible option, but instead acts as a basic benchmark: random allocation is what we might resort to if we knew nothing about the system state, and so any decent state-dependent policy should do better.

The second policy is the simple greedy policy: the controller never keeps a reserve, and whenever there is an asset to allocate, the controller puts it wherever results in the highest instantaneous reward rate. That is, the controller allocates the asset to task $l \in \arg\max_k R(\boldsymbol{x} + \mathbf{e}_k)$. This policy is in many ways the simplest sensible policy, and if we did not have to worry about failures and replenishment would in fact be optimal. We examined other variants of the greedy policy (such as using $l \in \arg\max_k R(\boldsymbol{x} + \mathbf{e}_k)/\mu_k$), but none were markedly different to the greedy policy and so we omit them.

The last policy we shall use for comparison is the optimal policy, which we compute by policy iteration (Puterman, 2014, Section 8.5). We provide pseudocode for our policy iteration algorithm in Appendix B. As discussed earlier, the state-space grows very quickly, and as such we are only able to calculate the optimal policy for small values of N and K.

The optimal policy is the only policy we consider that makes use of the reserves; all other policies immediately assign an asset after repair. While we did consider allowing the other policies to retain reserves, there is no natural way to do this, and the performance of the clever Whittle policy is sufficient to make further finesses unnecessary.

5. Numerical experiments

We now conduct numerical experiments to evaluate the performance of the policies defined above. We first re-parametrise the problem, so that the numerical experiments are more clear. We have five main inputs to our model: N, K, $g_k(x_k)$, $\lambda(x_k)$, and μ_k. We specify the reward functions $g_k(x_k)$ as follows. Recall that rewards are gained independently: $R(\boldsymbol{x}) = \sum_{k=1}^{K} g_k(x_k)$. In this section we focus on the following special form:

$$R(\boldsymbol{x}) = \sum_{k=1}^{K} \left(1 + \frac{Ak}{K}\right) g(x_k) \tag{41}$$

That is, we have one reward function $g(x)$, and one parameter $A > 0$ that specifies how quickly reward rates increase with k.

We assume there is no queuing for repairs; as soon as an asset fails it immediately begins the repair process. Therefore, $\lambda(x) = lx$ and we normalise $l = 1$ throughout. The failure rates take the form $\mu_k = Mm_k$, where M is a constant we vary to control the magnitude of the failure rate, and the set $(m_1, m_2, ..., m_K)$ is a sequence of values near 1; the crucial aspect is the ratios of the m_k to each other. Specifically we look at four cases for the form of $(m_1, m_2, ..., m_K)$

Case 1: Constant m_k: $m_k = 1$ for all $1 \le k \le K$
Case 2: m_k increasing with k: $m_k = 0.5 + \frac{k-1}{K-1}$
Case 3: m_k decreasing with k: $m_k = 1.5 - \frac{k-1}{K-1}$
Case 4: m_k oscillating in k, $m_k = 1.5$ if k odd, $m_k = 0.5$ if k is even.

Given the discussion above, we require the following six parameters to completely specify our problem: N, K, $g(x)$, A, M, m_k.

In our numerical experiments we examine all combinations of the parameters within the following limits:

- $N \in \{2, \dots, 10\}$
- $K \in \{2, \dots, 5\}$
- $g(x) \in \{1 - e^{x/5}, \quad \log(1+x), \quad \sqrt{x}, \quad \min(x, 2)\}$
- $A \in \{1, 2, 3, 4\}$
- $M \in \{0.1, 0.5, 1, 1.5, 2, 3, 5, 10\}$
- m_k: the four cases described above

These parameter combinations produce 18,432 different scenarios.

5.1. Policy evaluation

Owing to the continuous-time Markov process structure of the problem, we can compute the long-term time-average reward exactly for a fixed policy $\boldsymbol{u}$. We denote the long-term time-average reward $V^{\boldsymbol{u}}$ for policy $\boldsymbol{u}$. Any policy $\boldsymbol{u}$ induces a transition matrix $Q^{\boldsymbol{u}}$ on the state-space. We can find the stationary distribution $\pi^{\boldsymbol{u}}$ from the balance equations $\pi^{\boldsymbol{u}} Q^{\boldsymbol{u}} = 0$, with a normalization constraint to ensure that $\pi^{\boldsymbol{u}}$ is a probability distribution. Once we have $\pi^{\boldsymbol{u}}$, we compute the long-term time-average reward via $V^{\boldsymbol{u}} = \sum_{\boldsymbol{x} \in \mathcal{X}} R(\boldsymbol{x}) \pi^{\boldsymbol{u}}(\boldsymbol{x})$.

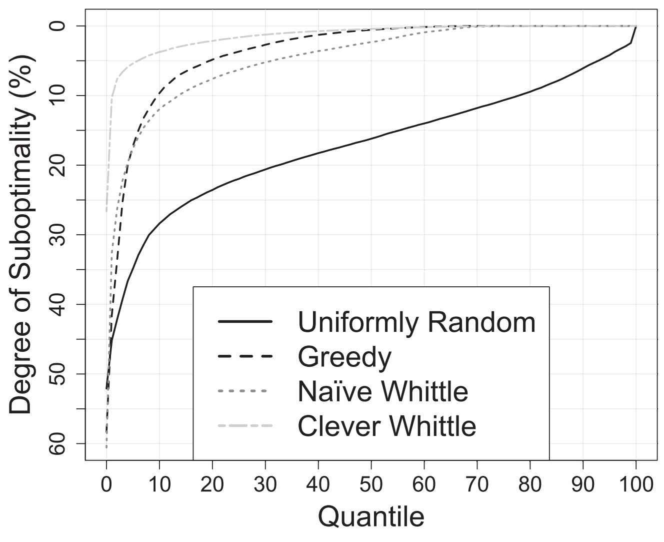


Fig. 4. Policy performance for each quantile.

5.2. Results

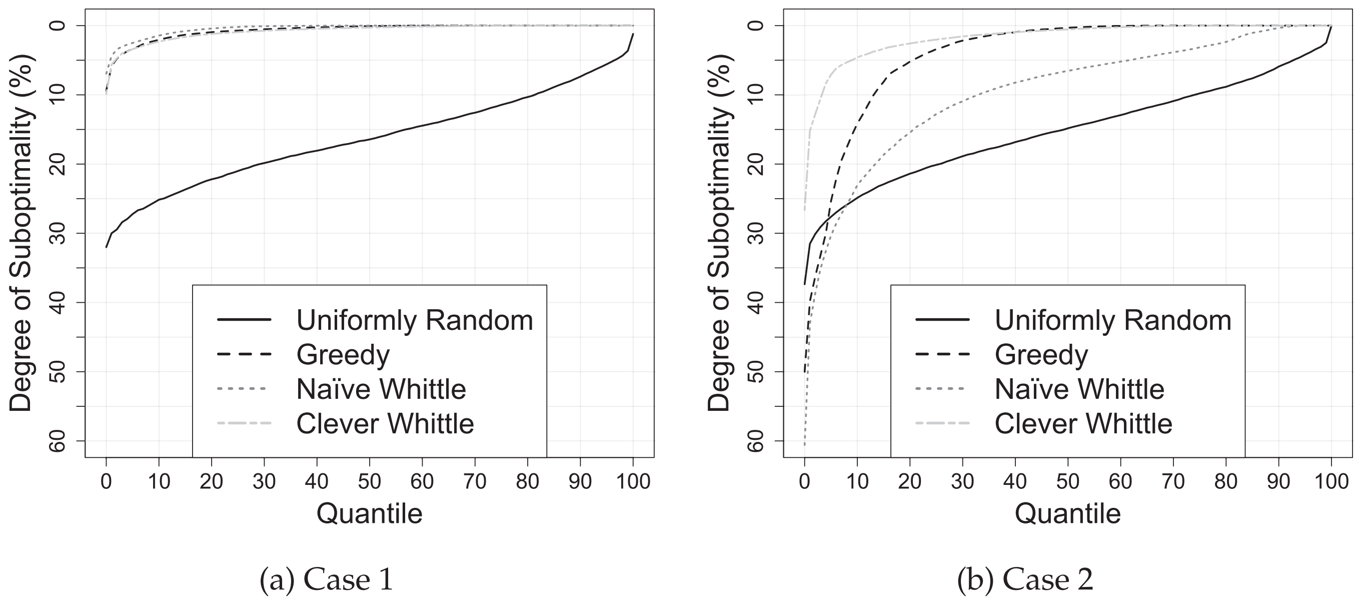
Our measure of performance is a policy's long-term time-average reward rate relative to the optimal, measured as the relative suboptimality gap of policy $\boldsymbol{u}$:

$$1 - \frac{V^{\boldsymbol{u}}}{V}$$

That is, for each set of parameters, we get a number between 0 and 1, which we report as a percentage, where 0% is identical to the optimal policy, and 100% equates to no reward at all. Note that in the figures, performance improves as we move up the y-axis.

Overall performance: The curves in Fig. 4 show the performance quantiles across all 18,432 parameter combinations. Table 1 gives a statistical summary for this case. Not surprisingly the uniformly random policy performs poorly relative to the other three. The clever Whittle index rule clearly dominates the other options, and is significantly more robust: even in the 5th percentile, it is at only 5.4% underperformance. The greedy policy performs better than the naïve Whittle index rule across most percentiles, but both struggle significantly for some cases. The median performance is 0.6% for greedy, 1.3 % for naïve Whittle, and 0.4% for clever Whittle.

Failure rates: The failure rate has a significant impact on the results. Recall we use two parameters to specify the failure rate $\mu_k = Mm_k$. M controls the magnitude of the failure rates, whereas m_k determines the form of the rates in k. Table 2 presents the 5th percentile performance for each of the 4 cases for m_k (constant, increasing, decreasing, oscillating) and Fig. 5 presents the quantile plots. Greedy performs reasonably well in Case 1 and Case 3 but not well in Case 2 and even worse in Case 4. The naïve Whittle index rule slightly dominates over greedy in Case 1 (even over the clever Whittle), and very significantly across lower quantiles in Case 4, but has a significantly worse performance in Case 2. Clever Whittle performs consistently well for all 4 cases, showing it is robust to changes in forms of failure rates.



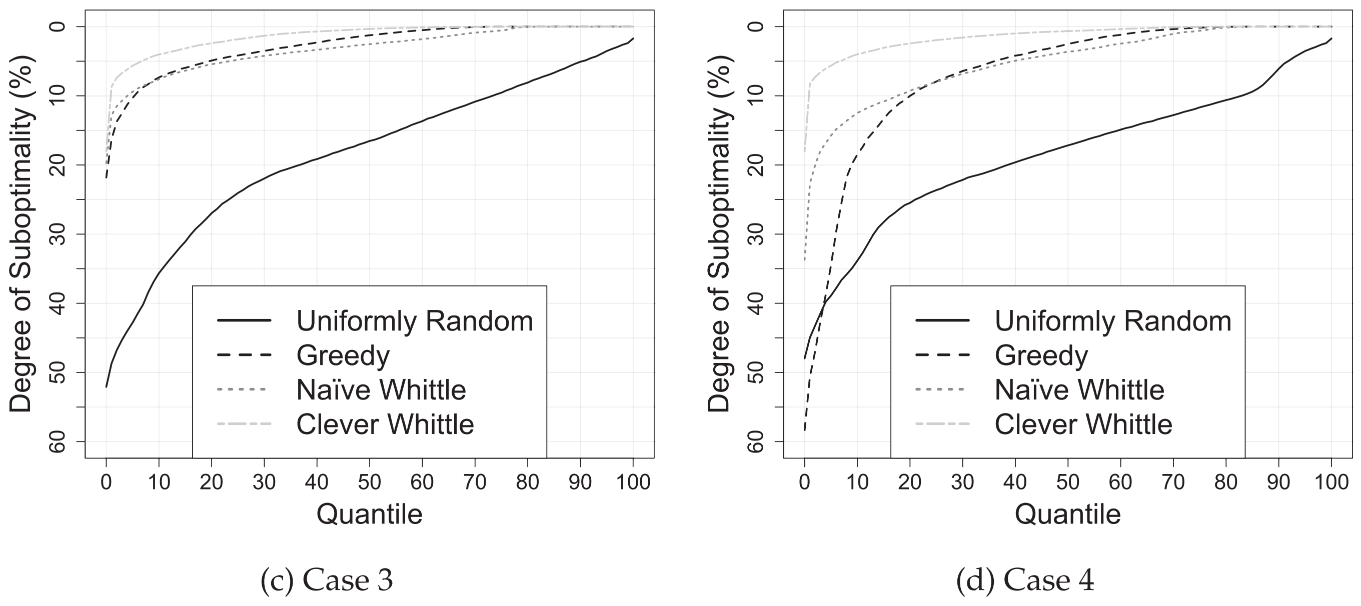
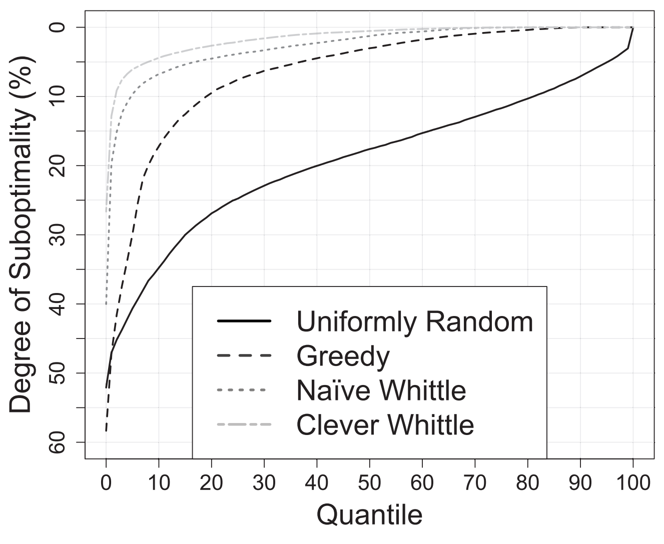
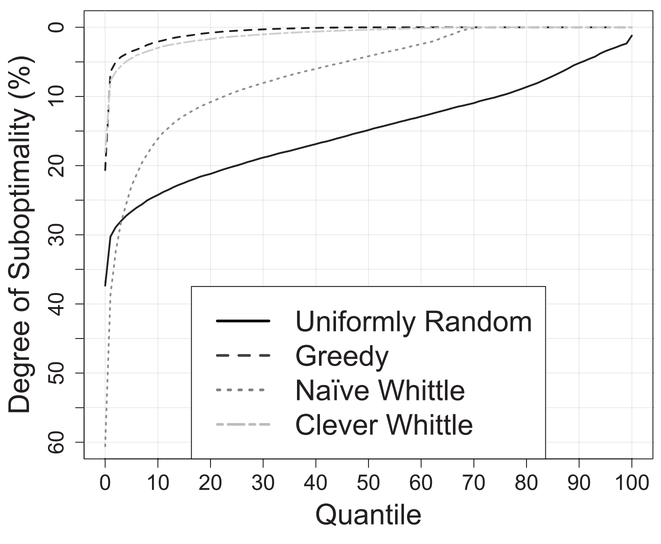


Fig. 5. Quantile plots, stratified by the failure rate forms.



(a) $M \leq 1$



(b) $M > 1$

Fig. 6. Quantile plots, stratified by the failure rate magnitude M.

Table 2
The 5th percentile for each policy, stratified by the failure rate forms.

Policy | Case	Case 1 (constant)	Case 2 (increasing)	Case 3 (decreasing)	Case 4 (oscillating)
Uniformly Random	27.2	27.6	42.8	38.9
Greedy	3.4	25.4	10.3	34.5
Naïve Whittle	2.5	30.3	9.4	15.7
Clever Whittle	3.4	7.1	5.6	5.6

We see similar behaviour in Table 3 and Fig. 6 as we vary the magnitude M of the failure rates. Recall the repair rate has the form $\lambda(x) = x$, so $M = 10$ ($M = 0.1$) corresponds to very large (small) failure rates relative to repair. For large M, greedy performs very well and naïve Whittle performs almost as poorly as the uniformly random policy. In these cases, assets spend most of the time performing tasks, with only short repair times. It is not surprising greedy does well as the greedy policy is optimal if the assets never fail. However for small M when assets fail quickly, the greedy performs almost as poorly as the uniformly random policy and the naïve Whittle performs well, even better than the clever Whittle. In this case the rapid barrage of failures produces a system that more closely mimics the open-system approximation with a constant arrival rate. Although the clever Whittle does not perform as well as either greedy or naïve Whittle in the extremes, it is again very robust across all values of M.

Use of the reserve: The reserve is used in 33% of the scenarios. In the most extreme cases, the system is in a state with an asset in reserve 25% of the time. However, overall the reserve component usually plays only a very minor role. Only 2% of the scenarios that optimally use the reserve are in a reserve state more than 10% of the time. The fact that our policies, which do not use reserves, perform so well illustrates that in general, the reserve does not serve a critical role. The reserve is more important for larger N and smaller K. The reserve is also used more frequently for larger M (greater failure rate relative to repair rate). In this case a decision maker might prefer to wait for another imminent failure before allocating an available asset to ensure the allocation is effective, because the next allocation opportunity may not be for a while.

Other Points: For the four reward function options, $g(x) = \min(x, 2)$ most often produces the worst performance. This is not surprising as none of our policies allows for reserves, and clearly it is sub-optimal to allocate more than two assets per task for this reward function. Consequently performance degrades for large N and small K for this reward function.

Finally, there is a slight degradation in performance of the policies as we increase the number of assets N. For the median, the clever Whittle's median relative suboptimality gap is 0.0% at $N = 2$ and increases to 0.9% at $N = 10$. The corresponding degradation at the 5th percentile for the clever Whittle index rule is from 2.2% to 6.4%.

6. Conclusion

We have formulated a model for the allocation of assets when the assets are subject to failure. We establish a solid theoretical grounding for the Whittle index rule and examine the performance of two Whittle index rule variants along with several other policies. From the numerical experiments, the clever Whittle index rule performs well and appears to be a reasonable and robust policy to generally use in practice.

There are several avenues for future research to improve our model. Rewards may not accumulate independently across the tasks. One might need a critical mass of assets at each task before

Table 3
The 5th percentile for each policy, stratified by the failure rate magnitude M.

Policy | M	10	2	1.43	1	0.5	0.33	0.2	0.1
Uniformly Random	28.6	26.9	25.8	25.1	26.4	28.9	32.1	46.6
Greedy	2.2	3.1	3.6	4.3	8.1	12.3	17.3	45.7
Naïve Whittle	27.0	24.5	21.5	19.4	14.0	11.0	8.0	5.0
Clever Whittle	4.2	4.3	4.6	4.5	4.5	5.1	6.0	11.2

the controller can gain positive rewards. The restless bandit model would not apply to a more complex reward function with dependencies across tasks; both the Whittle index rule and the greedy policy would need to be modified.

The Markovian assumptions, while unrealistic for some real-world applications, are not easy to discard. In practice it may be better to model failure times as uniform or deterministic: each asset has a certain level of charge, which depletes in a fairly predictable way. A possible intermediate approach would be the use of Erlang distributions, which are still simple enough to allow for effective calculations, while sufficiently complicated to model actual situations. One could also perform a simulation exercise to apply the policies formulated here for the Markovian case to a non-Markovian setting.

There might be delays between assignment and when an assets starts performing a task, such as travel times. Some of this delay can be absorbed into the repair process. We could also modify the reward function to account for the delay cost.

We mentioned in Section 4.2 the possibility of adding a reserve to the policies derived from the restless bandit model. This would best be done by adding a reserve to the restless bandit model, although there is no simple way of doing this. In particular the presence of a reserve links the different tasks, something wholly imcompatible with a restless bandit model.

The assumption that assets cannot be moved from one task to another may be reasonable in certain situations in practice (e.g. where the tasks are located too far away from each other or where there are few specialized assets), but the range of potential applications would greatly expand if asset relocation was allowed, possibly incorporating a switching cost and/or delay. Whittle index for such models has however not be studied yet; only its special case, so-called Gittins index, which does not apply in this context.

One last complication is rewards that are unknown *a priori*. The controller would have to learn about the rewards gained per state as she tries to stay in high-reward states. The framework here is analogous to a classic bandit problem (Gittins, Glazebrook, & Weber, 2011). Unlike a bandit problem, we have a complicated state-space, and the dynamics are not entirely under our control. This would be particularly relevant to environmental monitoring cases, where the controller might not initially know which locations are most important to observe.

Acknowledgments

We would like to thank the reviewers for their helpful and relevant comments, which have greatly improved this paper. This research was supported by an EPSRC Doctoral Training Grant, as part of the STOR-i Centre for Doctoral Training.

Appendix A. Proving the identity

We wish to prove that:

$$w(x) = \frac{\sum_{y=0}^{x} P_y(g(y+1) - g(y))}{\mu \sum_{y=0}^{x} P_y} \tag{42}$$

Begin with the expression for $w(x)$:

$$w(x) = \frac{\sum_{y=0}^{x+1} g(y)\pi_{x+1}(y) - \sum_{y=0}^{x} g(y)\pi_x(y)}{\Lambda(\pi_x(x) - \pi_{x+1}(x+1))} \tag{43}$$

Next we define

$$\rho = \frac{\Lambda}{\mu} \tag{44}$$

Recall from Eq. (21) that $P_x = \sum_{y=0}^{x} \rho^y/y!$, which is the normalising constant for $\pi_x(x)$. The second sum in the numerator of $w(x)$ can be written as:

$$\sum_{y=0}^{x} \pi_x(y)g(y) = \frac{1}{P_x}\left(\sum_{y=0}^{x} \frac{\rho^y}{y!} g(y)\right) \tag{45}$$

Take out a term $g(x)$ from the sum:

$$\sum_{y=0}^{x} \pi_x(y)g(y) = \frac{1}{P_x}\left(\left(\sum_{y=0}^{x} \frac{\rho^y}{y!}\right)g(x) + \sum_{y=0}^{x-1} \frac{\rho^y}{y!}(g(y) - g(x))\right) \tag{46}$$

Using the definition of P_x:

$$\sum_{y=0}^{x} \pi_x(y)g(y) = \frac{1}{P_x}\left(P_x g(x) + \sum_{y=0}^{x-1} \frac{\rho^y}{y!}(g(y) - g(x))\right) \tag{47}$$

We can repeat what we just did:

$$\sum_{y=0}^{x} \pi_x(y)g(y)$$
$$= \frac{1}{P_x}\left(P_x g(x) + \sum_{y=0}^{x-1} \frac{\rho^y}{y!}(g(x-1) - g(x)) + \sum_{y=0}^{x-1} \frac{\rho^y}{y!}(g(y) - g(x-1))\right) \tag{48}$$

Rewrite, noting that the last term of the second sum is zero:

$$= \frac{1}{P_x}\left(P_x g(x) + P_{x-1}(g(x-1) - g(x)) + \sum_{y=0}^{x-2} \frac{\rho^y}{y!}(g(y) - g(x-1))\right) \tag{49}$$

By doing this repeatedly, we can collapse the whole sum:

$$\sum_{y=0}^{x} \pi_x(y)g(y) = \frac{1}{P_x}\left(g(x)P_x + \sum_{y=0}^{x-1} P_y(g(y) - g(y+1))\right) \tag{50}$$

Now look at the numerator of the formula for $w(x)$. Using the expression just obtained, we can express it as:

$$\frac{1}{P_{x+1}}\left(g(x+1)P_{x+1} + \sum_{y=0}^{x} P_y(g(y) - g(y+1))\right)$$
$$- \frac{1}{P_x}\left(g(x)P_x + \sum_{y=0}^{x-1} P_y(g(y) - g(y+1))\right) \tag{51}$$

Joining the two sums together:

$$= \frac{P_x}{P_x}(g(x+1) - g(x)) + \frac{P_x(g(x) - g(x+1)}{P_{x+1}}$$
$$+ \left(\frac{1}{P_{x+1}} - \frac{1}{P_x}\right)\sum_{y=0}^{x-1} P_y(g(y) - g(y+1)) \tag{52}$$

The numerator is therefore equal to:

$$\frac{P_{x+1} - P_x}{P_{x+1}P_x}\left(\sum_{y=0}^{x} P_y(g(y+1) - g(y))\right) \tag{53}$$

For the denominator of $w(x)$, we can simplify directly:

$$\pi_x(x) - \pi_{x+1}(x+1) = \frac{P_x - P_{x-1}}{P_x} - \frac{P_{x+1} - P_x}{P_{x+1}} = \frac{P_x^2 - P_{x+1}P_{x-1}}{P_{x+1}P_x} \tag{54}$$

We can now combine the two expressions we have obtained to get a new form for $w(x)$, with a factor of $P_x P_{x+1}$ cancelling:

$$w(x) = \frac{1}{\Lambda}\frac{(P_{x+1} - P_x)\left(\sum_{y=0}^{x} P_y(g(y+1) - g(y))\right)}{P_x^2 - P_{x+1}P_{x-1}} \tag{55}$$

We can simplify the terms involving P_x by rearranging as follows:

$$\frac{(P_{x+1}-P_x)}{{P_x}^2-P_{x+1}P_{x-1}}=\frac{\rho^{x+1}/(x+1)!}{P_x^2-(P_x-\frac{\rho^x}{x!})(P_x+\frac{\rho^x+1}{(x+1)!})} \tag{56}$$

Multiply out the brackets on the bottom and cancel:

$$=\frac{\rho^{x+1}/(x+1)!}{P_x(\frac{\rho^x}{x!}-\frac{\rho^{x+1}}{(x+1)!})+\frac{\rho^x}{x!}\frac{\rho^{x+1}}{(x+1)!}} \tag{57}$$

Strip out a factor of $\rho^{x+1}/(x+1)!$ from top and bottom:

$$=\frac{1}{P_x(\frac{x+1}{\rho}-1)+\frac{\rho^x}{x!}}=\frac{1}{P_x\frac{x+1}{\rho}-P_x+\frac{\rho^x}{x!}}=\frac{1}{P_x\frac{x+1}{\rho}-P_{x-1}} \tag{58}$$

Simplifying this expression requires us to prove a lemma.

Lemma. $P_x\frac{x+1}{\rho}-P_{x-1}=\frac{1}{\rho}\sum_{y=0}^{x}P_y$.

Proof. We proceed by induction. For $x=1$ we substitute in $P_0=1$ and $P_1=1+\rho$:

$$(1+\rho)\frac{2}{\rho}-1=\frac{1}{\rho}(1+1+\rho) \tag{59}$$

$$\frac{2}{\rho}+2-1=\frac{1}{\rho}+\frac{1}{\rho}+1 \tag{60}$$

The two sides are equal, so we have our base case.

For the induction step, we assume the $x-1$ case, that:

$$P_{x-1}\frac{x}{\rho}-P_{x-2}=\frac{1}{\rho}\sum_{y=0}^{x-1}P_y \tag{61}$$

When we go to the x case, we must add the same to both sides; we must have that:

$$P_x\frac{x+1}{\rho}-P_{x-1}\frac{x}{\rho}-\frac{\rho^{x-1}}{(x-1)!}=\frac{1}{\rho}P_x \tag{62}$$

We can manipulate the L.H.S as follows, beginning by taking out a factor of ρ^{-1}:

$$\frac{1}{\rho}\left(\left(P_{x-1}+\frac{\rho^x}{x!}\right)(x+1)-P_{x-1}x-\frac{\rho^x}{(x-1)!}\right)=\frac{1}{\rho}P_x \tag{63}$$

Cancel the factor ρ^{-1}, and multiply out the brackets:

$$xP_{x-1}+P_{x-1}+\frac{x\rho^x}{x!}+\frac{\rho^x}{x!}-P_{x-1}x-\frac{\rho^x}{(x-1)!}=P_x \tag{64}$$

$$P_{x-1}+\frac{\rho^x}{x!}+\frac{\rho^x}{(x-1)!}-\frac{\rho^x}{(x-1)!}=P_x \tag{65}$$

$$P_{x-1}+\frac{\rho^x}{x!}=P_x \tag{66}$$

We have hence established the required identity.

We can now use this result and rewrite w_x as:

$$w(x)=\frac{1}{\mu}\frac{\sum_{y=0}^{x}P_y(g(y+1)-g(y))}{\sum_{y=0}^{x}P_y} \tag{67}$$

This is the required identity. □

Appendix B. Pseudocode for policy iteration

Throughout, **0** is the vector of all zeros, and e_i is the ith basis vector, both in the appropriate number of dimensions. Comments are indicated by two hashes: ##.

U is the policy, represented as a vector of the controls, indexed by the states; $U(x)=a$ is the control applied in state x. max_iterations is there in case the iteration takes too long to converge, and is a hard cap on the number of times the central loop can run.

```
PolicyIteration := function(U,max_iterations){
## We need to set up a few things
## First, the size of the state-space
$|X| = \begin{pmatrix} N+K+1 \\ K+1 \end{pmatrix}$
## The relative value vector
$\phi = 0$
## Uniformisation constant
$B = N * max(\mu_i) + \lambda(N)$
## Iteration cap
iteration_number=0
while(iteration_number < max_iterations){
    ## Store the old controls
    $old\_U = U$
    $A = \mathbf{0}\mathbf{0}^T$ ## I.e. an all zeroes matrix
    $r = \mathbf{0}$
    ## Evaluate the current control
    for($x \in X$){
        $a = U(x)$        $r(x) = R(x+a)/B$
        ## Self-transition, which returns us to $x$ (this is fine
        ## because the problem is
        ## memoryless and so we'll just apply the same control
        ## again)
        $A[x,x] = -\lambda((x+a)_{K+1}))/B - (\sum_{k=1}^{K} \mu_k(x+a)_k/B)$
        ## Repair transition
        if($(x+a)_{K+1} > 0$){
            $p_{xy} = \lambda((x+a)_{K+1})/B$
            $y = x+a-e_{K+1}+e_{K+2}$
            $A[x,y] = -p_{xy}$
        }
        ## Failure transitions
        for($k \in K$ s.t. $(x+a)_k > 0$){
            $p_{xy} = \mu_k(x+a)_k/B$
            $y = x+a-e_k+e_{K+1}$
            $A[x,y] = -p_{xy}$
        }
        ## Average reward
        $A[x,|X|+1] = 1$
    }
    ## Fixing condition: $\phi[1] = 0$
    $A[|X|+1,1] = 1$
    $r[|X|+1+1] = 0$
    $\phi = A^{-1}r$
    ## Now solve for the optimal controls
    for($x \in X$ s.t. $x_{K+2} > 0$){
        $val = \mathbf{0}$, of length $|U|$
        for($a \in U(x)$){
            ## Reward and self-transition
            $val[a]=R(x+a)+\phi(x)(B-\lambda((x+a)_{K+1})-\sum_{k=1}^{K}\mu_k(x+a)_k)$
            ## Repair transition
            if($x_{K+1} > 0$){
                $q_{xy} = \lambda((x+a)_{K+1})/B$
                $y = x+a-e_{K+1}+e_{K+2}$
                $val[a] = val[a]+q_{xy}\phi[y]$
            }
            ## Failure transitions
            for($k \in K$ s.t. $(x+a)_k > 0$){
                $q_{xy} = \mu_k(x+a)_k$
                $y = x+a-e_k+e_{K+1}$
                $val[a] = val[a]+q_{xy}\phi[y]$
            }
        }
```

```
            ## Pick the best control
            $U(x) = \arg\max val$
        }
        ## Check the termination condition
        if($old\_U == U$){
        $return(U)$
        }
        iteration_number = iteration_number + 1
    }
    print(”Maximum Iterations Reached”)
    return(U)
}
```

References

Borkar, V. S., & Pattathil, S. (2017). Whittle indexability in egalitarian processor sharing systems. *Annals of Operations Research, Online First*.

Boyacı, B., Zografos, K. G., & Geroliminis, N. (2015). An optimization framework for the development of efficient one-way car-sharing systems. *European Journal of Operational Research, 240*(3), 718–733.

Casbeer, D. W., Beard, R. W., McLain, T. W., Li, S.-M., & Mehra, R. K. (2005). Forest fire monitoring with multiple small UAVs. In *Proceedings of the american control conference*.

Cook, K. L. B. (2007). The silent force multiplier: The history and role of UAVs in warfare. In *Proceedings of the IEEE aerospace conference* (pp. 1–7).

Doshi, B. T. (1986). Queueing systems with vacations: A survey. *Queueing Systems, 1*, 29–66.

Dufour, F., & Piunovskiy, A. B. (2015). Impulsive control for continuous-time Markov decision processes. *Advances in Applied Probability, 47*(1), 106–127.

Federgruen, A., & So, K. C. (1991). Optimality of threshold policies in single-server queueing systems with server vacations. *Advances in Applied Probability, 23*(2), 388–405.

Gittins, J. C. (1979). Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological), 41*(2), 148–164.

Gittins, J. C., Glazebrook, K., & Weber, R. (2011). *Multi-armed Bandit Allocation Indices*. John Wiley & Sons.

Glazebrook, K., Kirkbride, C., & Ouenniche, J. (2009). Index policies for the admission control and routing of impatient customers to heterogeneous service stations. *Operations Research, 57*, 975–989.

Gonzalez, L. F., Montes, G. A., Puig, E., Johnson, S., Mengersen, K., & Gaston, K. J. (2016). Unmanned aerial vehicles (UAVs) and artificial intelligence revolutionizing wildlife monitoring and conservation. *Sensors, 16*(1), 97.

Harchol-Balter, M. (2013). *Performance Modeling and Design of Computer Systems: Queueing Theory in Action*. Cambridge University Press.

Hyytiä, E. (2013). Optimal routing of fixed size jobs to two parallel servers. *INFOR: Information Systems and Operational Research, 51*(4), 215–224.

Ke, J.-C. (2003). The optimal control of an M/G/1 queueing system with server vacations, startup and breakdowns. *Computers & Industrial Engineering, 44*(4), 567–579.

Kerzner, H. (2017). *Project Management: A Systems Approach to Planning, Scheduling, and Controlling* (10th ed.). John Wiley & Sons.

Lin, K., Atkinson, M., & Glazebrook, K. (2014). Optimal patrol to uncover threats in time when detection is imperfect. *Naval Research Logistics, 61*(8), 557–576.

Nigam, N. (2014). The multiple unmanned air vehicle persistent surveillance problem: A review. *Machines, 2*(1), 13–72.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. John Wiley & Sons.

Puterman, M. L. (2014). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons.

Schuijbroek, J., Hampshire, R. C., & Van Hoeve, W.-J. (2017). Inventory rebalancing and vehicle routing in bike sharing systems. *European Journal of Operational Research, 257*(3), 992–1004.

Stidham, S., & Weber, R. (1993). A survey of Markov decision models for control of networks of queues. *Queueing systems, 13*(1–3), 291–314.

Waharte, S., & Trigoni, N. (2010). Supporting search and rescue operations with UAVs. In *Proceedings of the International conference on emerging security technologies (est)* (pp. 142–147).

Whittle, P. (1988). Restless bandits: Activity allocation in a changing world. *Journal of Applied Probability, 25*(A), 287–298.