Abstract

We consider the problem \((J, \Gamma)\) of allocating a single machine to the stochastic tasks in \(J\) in such a way that precedence constraints \(\Gamma\) are respected. If we have rewards which are discounted and additive then the problem of determining an optimal policy for scheduling in the class of fully preemptive policies can be formulated as a discounted Markov decision process (MDP). Policies are developed by utilising a principle of forwards induction (FI). Such policies may be thought of as quasi-myopic in that they make choices which maximise a natural measure of the reward rate currently available. A condition is given which is (necessary and) sufficient for the optimality of FI policies and which will be satisfied when \(\Gamma = \{\text{out-forest}\}\). The notion of reward rate used to develop FI policies can also be used to develop performance bounds for general scheduling policies. These bounds can be used to make probabilistic statements about heuristics (i.e. for randomly chosen \((J, \Gamma)\)). The FI approach can also be used to develop policies for general discounted MDPs. Performance bounds are available which may be used to make probabilistic statements about the performance of FI policies in more complex scheduling environments where optimality results are not available.

1. Introduction

A single machine is available to process a set \(J\) of jobs in a way which respects partial order \(\Gamma\). Time evolves in discrete steps and the decision epochs are the natural numbers \(\mathbb{N}\) or some subset thereof. At each decision epoch \(t\), the machine is allocated to one of the jobs in \(J\) which has yet to complete. This decision will remain in force until the following epoch. \(\Gamma\) is a subset of \(J \times J\), where \((i,j) \in \Gamma\) denotes the requirement that task \(i\) be completed before the processing of \(j\) can begin. Use the shorthand \((J, \Gamma)\) to denote such a model.

We write \(X(t) = \{X_1(t), X_2(t), \ldots, X_N(t)\}\) for the state of the system at \(t \in \mathbb{N}\), where \(N = |J|\) and \(X_j(t)\) is the state of job \(j\) at \(t\). Should a decision be made to process \(j\) at decision epoch \(t\) then \(X_j(\cdot)\) evolves as a Markov chain, with transitions at \(t + 1, t + 2, \ldots\) until the next decision epoch. The state space \(\Omega_j\) for job \(j\) is quite general and contains completion set \(\omega_j\) as a subset. Job \(j\) is complete as soon as its state enters \(\omega_j\).
A policy is a rule for allocating the machine to a currently available job, i.e. a job which has yet to complete, but all of whose predecessors under \( I \) are complete. Any such rule may take account of the entire history of the process to date. We usually restrict discussion to some subclass of policies of interest. Important subclasses of policies are the following.

(i) **Static list policies:** At \( t = 0 \), a feasible permutation is chosen, specifying the order in which the jobs are to be processed. No subsequent changes are allowed. All jobs are processed through to completion.

(ii) **Nonpreemptive (dynamic) policies:** This is the class of policies obtained from the requirement that decision epochs occur only at job completion times. Static list policies are included as an important subclass. Under both (i) and (ii), once a job’s processing has begun, the machine is committed to the job until its completion.

(iii) **Preemptive (dynamic) policies:** Here, all nonnegative integer time points which precede the last job completion are decision epochs. Hence, the machine is able to switch between currently available jobs in an unrestricted way.

(iv) **(Dynamic) policies with limited preemptions:** There are a variety of ways of characterising classes of policies which lie somewhere between (ii) and (iii), i.e. (sub-)classes of the preemptive policies which contain the nonpreemptive policies as a (sub-)class.

(v) **Single visit policies:** In some areas of application where the scheduling model has \( I = \emptyset \), \( \omega_j = \emptyset \), \( 1 \leq j \leq N \) (i.e. there is no notion of job completion and no precedence constraints), it makes sense to consider policies for which the machine has a single period of processing each job – the duration of \( j \)'s processing to be determined by some stopping time defined on the process \( X_j(\cdot) \). Hence, a single visit policy will be determined by a feasible permutation of \( J \) together with a vector of such stopping times.

Rewards are earned as jobs are processed. Should job \( j \) be processed during \([t, t+1)\), an expected reward \( \alpha^t R_j\{X_j(t)\} \) is earned from that processing. The discount rate \( \alpha \) is in \([0, 1)\) and each reward function \( R_j: \Omega_j \rightarrow \mathbb{R}^+ \) is bounded, \( 1 \leq j \leq N \).

An **optimal policy** (within the specified policy class) maximises the total expected reward earned. Write \( \pi \) for a policy and \( R_{\pi}(x) \) for the total expected reward earned by \( \pi \) from initial state \( x \in \times_{j=1}^N \Omega_j \). We write

\[
R_{\pi}(x) = E_{\pi} \left[ \sum_{t=0}^{\infty} \alpha^t R\{\pi, X(t)\} \bigg| X(0) = x \right],
\]

where \( E_{\pi} \) is an expectation taken over realisations of the system under policy \( \pi \) and \( R\{\pi, X(t)\} \) is the reward earned by \( \pi \) at time \( t \).

With this reward structure, the problem of finding optimal policies within the class of static list policies is equivalent to the problem of finding optimal policies within the class of nonpreemptive (dynamic) policies. Since both are equivalent to deterministic
scheduling problems, relatively little emphasis will be placed on these policy classes in this account. Examples of stochastic scheduling models where the distinction between nonpreemptive and static list policies is crucial may be found in [7, 26]. We shall concentrate here primarily on the class of preemptive (dynamic) policies.

An advantage of the above reward structure (in addition to the flexibility it offers in modelling) is that the problem of finding an optimal preemptive policy is a discounted Markov decision problem, as discussed by Ross [39]. Standard theory ensures the existence of an optimal (preemptive) policy which is deterministic, stationary and Markov. We shall assume that our (preemptive) policies have these properties without further comment. We may, also without further comment, restrict attention to nonidling policies. Plainly, the option of inserting idle time may be represented by a zero reward, single state job. Should such an option ever be optimal, it will continue thereafter to be optimal. This yields a contradiction, since all rewards are nonnegative and no rewards are earned by the idleness option.

Many scheduling problems defined with respect to policies allowing limited preemptions can be viewed as semi-Markov versions of equivalent problems defined for the class of fully preemptive policies. The development in Sections 2 and 3 permits of trivial extension from the Markovian models discussed to equivalent semi-Markov ones. In this way, we may deduce extensions of the main results to classes of policies under a regime of limited preemption; see [17] for more details. The problem of finding optimal single visit policies is genuinely distinct, although we are able to make use of the ideas and methodologies used in the fully preemptive case; see [3] for an account.

We shall proceed as follows. In Section 2, we shall describe a class of forwards induction policies for \((J, F)\). These may be thought of as quasi-myopic in that they make choices which maximise the currently available reward rate (defined appropriately). If \(F\) is an out-tree, forwards induction policies are optimal in the class of preemptive (dynamic) policies. If \(F\) is general, nonpreemptive forwards induction policies are optimal in the class of nonpreemptive (dynamic) policies. These seem to be the best results available. Ideas related to the computation of forwards induction policies are described and some examples given. Major contributions to the ideas and methodologies in this account are to be found in [9, 10, 12, 24, 25, 30, 31, 33, 44, 45]. A short account of single visit policies is also given.

The notion of reward rate alluded to in the previous paragraph may also be used to develop performance bounds for general policies for \((J, F)\). This is discussed in Section 3. Such bounds are useful even when we have access to optimality results via the forwards induction principle, since forwards induction policies may be difficult to construct and/or apply. The development of these bounds goes back to Glazebrook [13]. Subsequent developments may be found in [10, 14, 19, 21, 22, 31]. These bounds have been used primarily for policy evaluation and sensitivity analysis for fixed \((J, F)\) (see [18]). However, there is no reason in principle why they should not be used as the basis for making probabilistic statements about the performance of heuristics (i.e. for randomly chosen \((J, F)\)). An example is given.
We can use the notion of forwards induction to develop policies for general discounted Markov decision processes. This is discussed in Section 4. A performance bound for such policies due to Glazebrook and Gittins [25] is presented. It is used to provide a probabilistic analysis of the forwards induction heuristic in a complex scheduling environment where optimality results are not available.

While the ideas and methodologies described in the paper have exerted a considerable influence on the development of stochastic scheduling, many important contributions have been made which fall outside this framework. See, for example, the contributions in [6] and the recent survey by Righter [37].

2. Forwards induction and Gittins indexation

Consider the (deterministic, stationary and Markov) preemptive policy \( \pi \) applied to \((J, F)\). Suppose that \( \tau > 0 \) is a positive-valued stopping time on this process and that \( X(0) = x \in \times_{j=1}^{N} \Omega_j \) is the initial state. Write \( R_{\pi, \tau}(x) \) for the total expected reward earned by \( \pi \) during \([0, \tau)\), i.e.

\[
R_{\pi, \tau}(x) = E_{\pi} \left[ \sum_{t=0}^{\tau-1} x^t R_{\pi, \tau}(x, X(t)) \mid X(0) = x \right]
\]

and

\[
G_{\pi, \tau}(x) = R_{\pi, \tau}(x) \left[ 1 - E_{\pi}(x^t \mid X(0) = x) \right]^{-1}
\]

for the corresponding reward rate.

**Definition 1.** The Gittins index for \((J, F)\), \( G: \times_{j=1}^{N} \Omega_j \to \mathbb{R}^+ \) is defined by

\[
G(x) = \sup_{\pi, \tau} G_{\pi, \tau}(x), \quad x \in \times_{j=1}^{N} \Omega_j.
\]

Hence, a Gittins index is a maximal reward rate. We shall see later how the structure of \((J, F)\) may sometimes be exploited to simplify the calculation of these indices and induce "separability" results. Two immediate and important questions related to Definition 1 are:

1. Is the supremum on the r.h.s. of (2.2) attained?
2. How can \( G(x) \) be calculated?

In answer to these questions, Katehakis and Veinott [31] consider the "restart in \( x \)" problem, modelled as a discounted MDP as follows: at each decision epoch \( t \in \mathbb{N} \), either take one of the actions available in \((J, F)\) in state \( X(t) \) or take the "restart" action followed by one of the actions available in \((J, F)\) in initial state \( x \). The effects of actions in terms of transitions/rewards are as in \((J, F)\). Hence, the "restart in \( x \)" problem modifies the original scheduling problem \((J, F)\) by allowing the decision maker to reset the problem in its initial state at any decision epoch.
By the standard theory of discounted MDPs, there exists a deterministic, stationary and Markov policy for the “restart in x” problem. Such a policy may be expressed as a pair \((\pi, \tau)\) where \(\pi\) is a policy for \((J, \Gamma)\) and \(\tau\) is a positive-valued stopping time on the process under \(\pi\) (indicating when the restart action should be taken). Write \(\tilde{R}_{\pi, \tau}(x)\) for the expected reward earned during \([0, \infty)\) by policy \((\pi, \tau)\) for the “restart in x” problem. Plainly, we have

\[
\tilde{R}_{\pi, \tau}(x) = R_{\pi, \tau}(x) = \sum_{n=0}^{\infty} [E_{\pi}\{\alpha^n | X(0) = x\}] = G_{\pi, \tau}(x).
\]

Questions (1) and (2) posed following Definition 1 may now be answered. The supremum on the r.h.s. of (2.2) is attained by a pair \((\tilde{\pi}, \tilde{\tau})\) yielding an optimal policy for the “restart in x” problem. The resulting value function is \(G(x)\) which may be obtained by applying the standard iterative approaches for discounted MDPs.

In order to say something about the pair \((\tilde{\pi}, \tilde{\tau})\) attaining the supremum in (2.2) consider now \((J, \Gamma)\) under some prespecified policy \(\pi\).

**Definition 2.** The Gittins index for \((J, \Gamma)\) under \(\pi\), \(G_{\pi} : \times_{j=1}^{N} \Omega_j \rightarrow R^+\) is defined by

\[
G_{\pi}(x) = \sup_{\tilde{\tau}} G_{\pi, \tilde{\tau}}(x), \quad x \in \times_{j=1}^{N} \Omega_j.
\]

Write \(\tau_{\pi}(x)\) for any stopping time attaining the supremum in (2.4). Nash [36] obtained the characterisation: \(\tau_{\pi}(x)\) is a stopping time defined on \((J, \Gamma)\) under \(\pi\) by

\[
\tau_{\pi}(x) = \inf\{t; t > 0 \text{ and } G_{\pi}\{X(t)\} < G_{\pi}(x)\}.
\]

The inequality < in (2.5) may be replaced by \(\leq\). That apart, Eq. (2.5) specifies \(\tau_{\pi}(x)\) uniquely. Hence if \((\tilde{\pi}, \tilde{\tau})\) attains the supremum in (2.2) then \(\tilde{\tau} = \tau_{\pi}(x)\). In fact, from Theorem 3.4(iii) of Gittins [10] we know that \(\tilde{\tau}\) is a stopping time defined on \((J, \Gamma)\) under \(\tilde{\pi}\) by

\[
\tilde{\tau} = \inf\{t; t > 0 \text{ and } G\{X(t)\} < G(x)\}.
\]

An alternative characterisation of Gittins indices based around the notion of retirement is due to Whittle [44]. To develop this idea, denote by \((J, \Gamma, M)\) the scheduling problem \((J, \Gamma)\) with retirement reward \(M\) as follows: at each decision epoch \(t \in \mathbb{N}\) either take one of the actions available in \((J, \Gamma)\) in state \(X(t)\) or take the retirement action. Should the retirement action be taken at \(t\), then \(X(t + 1) = X(t)\) and a positive reward \(x'M(1 - x)\) is received.

Denote by \(R_\pi(x, M)\) the total expected reward obtained when actions from \((J, \Gamma)\) are chosen according to policy \(\pi\) and retirement is taken optimally. Also write

\[
R(x, M) = \sup_{\pi} R_\pi(x, M).
\]
It is not difficult to show, from the definitions of the quantities involved, that
\[ G(x) = \inf \{ M; R(x, M) = M \}. \]
In this sense we have the Gittins index as an equivalent retirement reward. The following notion is due to Whittle [44].

**Definition 3.** Optimal policy \( \hat{\pi} \) for \((J, \Gamma)\) is dominating if
\[ R_{\hat{\pi}}(x, M) = R(x, M), \quad x \in \times_{j=1}^{N} \Omega_j, \quad M \geq 0. \]

Lemma 1 may now be easily inferred.

**Lemma 1.** If a dominating policy \( \hat{\pi} \) exists, then the supremum in (2.2) is attained by taking \( \pi = \hat{\pi} \) and \( \tau = \tau_{\hat{\pi}}(x) \) for all \( x \in \times_{j=1}^{N} \Omega_j \).

One final preparatory idea before using reward rates/Gittins indices to construct policies concerns the decomposition of the digraph representing \((J, \Gamma)\) into disjoint components. Suppose that there are \( m \) of the latter. We write \((J_i, \Gamma_i)\) for the stochastic scheduling problem obtained upon restriction (in the obvious way) to the \( i \)th component. We shall use superscript \( i \) to denote component \( i \) in the notation, e.g. \( G^i \) and \( G_{x, \tau} \), respectively, for the Gittins index and reward rate determined by \((\pi, \tau)\) for \((J_i, \Gamma_i)\), \( 1 \leq i \leq m \). Similarly, we use the notation \( x^i = \{x_j, j \in J_i\} \) for the state of component \( i \).

The following is a simple consequence of Lemma 3.12 in Gittins [10].

**Lemma 2.**
\[ G(x) = \max_{1 \leq i \leq m} G^i(x^i), \quad x \in \times_{j=1}^{N} \Omega_j. \]

Lemma 2 offers the prospect of a significant reduction in computational effort required to obtain \( G(x) \). Plainly, we may now substitute the \( m \) “restart in \( x \)” problems corresponding to \((J_i, \Gamma_i)\), \( 1 \leq i \leq m \), for the original “restart in \( x \)” problem used to compute \( G(x) \). According to Lemma 2, \( G(x) \) will then be the maximum of the \( m \) values \( G^i(x^i) \) thus obtained.

We are now ready to construct a forward induction (FI) policy \( \pi_G \) for \((J, \Gamma)\) as follows: at time 0 choose policy \( \hat{\pi}_1 \) and stopping time \( \hat{\tau}_1 \) (on \((J, \Gamma)\) under \( \hat{\pi}_1 \)) to attain the Gittins index \( G\{X(0)\} \). The FI policy constructed by this procedure implements \( \hat{\pi}_1 \) up to \( \hat{\tau}_1 \). The state \( X(\hat{\tau}_1) \) is observed and a new policy/stopping time pair \((\hat{\pi}_2, \hat{\tau}_2)\) is chosen to attain \( G\{X(\hat{\tau}_1)\} \). Policy \( \hat{\pi}_2 \) is then implemented during \([\hat{\tau}_1, \hat{\tau}_1 + \hat{\tau}_2)\), and so on. We develop a stochastic sequence \( \{(\hat{\pi}_n, \hat{\tau}_n), n \geq 1\} \) of such policy/stopping time pairs. Write \( \hat{s}_k = \sum_{r=1}^{k} \hat{\tau}_r \). Formally, we have the following definition.
Definition 4. \( \{ (\hat{\pi}_n, \hat{\tau}_n), n \geq 1 \} \) is a forwards induction (FI) sequence for \((J, \Gamma)\) if

(i) \( G_{\hat{\pi}_n, \hat{\tau}_n}(X(0)) = G(X(0)) \);

(ii) each \( \hat{\tau}_n \) is a positive-valued stopping time on \((J, \Gamma)\) under policy \( \hat{\pi}_n \) from initial state \( X(\hat{\tau}_{n-1}) \). The pair \( (\hat{\pi}_n, \hat{\tau}_n) \) is chosen such that

\[ G_{\hat{\pi}_n, \hat{\tau}_n}(X(\hat{\tau}_{n-1})) = G(X(\hat{\tau}_{n-1})); \]

(iii) in (ii) above the state \( X(\hat{\tau}_{n-1}) \) is the result of applying FI policy \( \pi_G \) to \((J, \Gamma)\) during \([0, \hat{\tau}_{n-1})\) from initial state \( X(0) \), this policy being such that

\[ \pi_G(X(t)) = \hat{\pi}_m(X(t)), \quad \hat{\tau}_{m-1} \leq t < \hat{\tau}_m, \quad m \geq 1. \]

The following facts about FI policies may easily be inferred from the preceding material:

(1) We may assume that at stage \( n \) of an FI policy (i.e. during \([\hat{\tau}_{n-1}, \hat{\tau}_n)\)) the policy \( \hat{\pi}_n \) in force chooses to process from just one of the components in \((J, \Gamma)\) in state \( X(\hat{\tau}_{n-1}) \). If, further, each such component has a dominating policy then by Lemma 1, \( \hat{\pi}_n \) will choose actions according to the dominating policy of the chosen component.

(2) From (2.6), stopping time \( \hat{\tau}_n \) may be expressed as

\[ \hat{\tau}_n = \inf\{t; t > \hat{\tau}_{n-1} \text{ and } G(X(t)) < G(X(\hat{\tau}_{n-1}))\}. \quad (2.7) \]

(3) From (2.7) it is immediate that the stochastic sequence of Gittins indices \([G(X(\hat{\tau}_n)), n \in \mathbb{N}]\) is decreasing almost surely.

(4) \( \Gamma = \emptyset \) is an important special case for which FI policies take a very simple form. Firstly, note that since \((J, \Gamma_i) \equiv \{i\}\) is now a single job then (trivially) the policy "process job \( i \)" is dominating. Hence, from (1) throughout stage \( n \) of an FI policy, the machine processes (only) job \( k \), where

\[ G_k(X_k(\hat{\tau}_{n-1})) = \max_{1 \leq j < N} G_j(X_j(\hat{\tau}_{n-1})). \quad (2.8) \]

In (2.8), \( G_k \) is a job-specific Gittins index for \( k \). Note that \( i, j \) and \( k \) appearing as subscripts will always denote individual jobs. In (2.8) and elsewhere we adopt the convention that a completed job (and, indeed, any set of completed jobs) has Gittins index 0. Since from Lemma 1 and (2.5) we have that

(a) \( G(X(\hat{\tau}_{n-1})) = G_k(X_k(\hat{\tau}_{n-1})) \) for some \( k \);

(b) \( G(x) = \max_{1 \leq j < N} G_j(x_j), x \in \times_{j=1}^\infty \Omega_j \);

(c) \( G_k(X_k(\hat{\tau}_{n-1})) \) is attained by the stopping time defined on the evolution of job \( k \) by

\[ \inf\{t; t > \hat{\tau}_{n-1} \text{ and } G_k(X_k(t)) < G_k(X_k(\hat{\tau}_{n-1}))\}, \]

we see that when \( \Gamma = \emptyset \) an FI policy chooses to process a job with maximal job-specific index at every decision epoch \( t \in \mathbb{N} \). Gittins [10] calls such a policy an index policy.
Theorem 3. **Index policies are optimal in the class of preemptive (dynamic) policies for \((J,0)\).**

To prove Theorem 3, Gittins and Jones [11] used an argument based on pairwise interchanges. Whittle [44] obtained a proof based on demonstrating that the value function resulting from an index policy satisfies the dynamic programming optimality equations. Weber [42] has recently given a rather simpler proof.

We now describe job-specific indices \(G_j\) for two rather different examples.

**Example 1** ("simple" job \(j\)). Assume that job \(j\) has processing requirement \(P_j\), a random variable taking values in the positive integers with completion rate \(\rho_j: \mathbb{N} \to [0, 1]\) defined by

\[
\rho_j(x_j) = p(P_j = x_j + 1 | P_j > x_j).
\]

Should job \(j\) complete at time \(t\), a reward \(\alpha r_j t\) is earned where \(r_j\) is a positive constant. No other rewards are earned by \(j\).

An adequate state description for such a simple job \(j\) is \((x_j, I_j)\) where \(x_j\) is the elapsed processing of \(j\) and \(I_j\) an indicator variable with the value 1 denoting completion. It is not difficult to see that, taking \(X_j(0) = (x_j, 0)\), a stationary stopping time defined on the evolution of job \(j\) must have the form "stop as soon as the elapsed processing is \(x_j + s\) or as soon as job \(j\) is complete, whichever occurs first". Hence, there is a 1–1 correspondence between stationary stopping times and \(\mathbb{Z}^+\). Invoking the notion of a Gittins index as a maximal reward rate we easily obtain

\[
G_j(x_j, 0) = \sup_{s \in \mathbb{Z}^+} \left[ \frac{r_j \sum_{t=1}^{s} \alpha^t \rho_j(x_j + t - 1) \prod_{u=0}^{t-2} (1 - \rho_j(x_j + u))}{(1 - \alpha) \sum_{t=0}^{s} \alpha^t \prod_{u=0}^{t-1} (1 - \rho_j(x_j + u))} \right],
\]

\(x_j \in \mathbb{N}\). \hspace{1cm} (2.9)

Note that if \(\rho_j\) is decreasing, the supremum in (2.9) is attained at \(s = 1\) yielding the myopic index \(\alpha x_j r_j (1 - x)\). Since this index is itself decreasing we would expect that optimal policies will involve a good deal of switching to and from job \(j\). If \(\rho_j\) is increasing the supremum in (2.10) is attained in the limit \(s \to \infty\) to obtain the index

\[
G_j(x_j, 0) = r_j \{E(\alpha^{P_j-x_j} | P_j > x_j) \{1 - E(\alpha^{P_j-x_j} | P_j > x_j)\}^{-1}, \ x_j \in \mathbb{N}. \hspace{1cm} (2.10)
\]

This index is itself increasing, ensuring that job \(j\) is processed nonpreemptively in an optimal policy.

**Example 2** (a search problem). \(N\) boxes each contain an unknown number of objects. Distribution \(\Pi_j\) summarises prior beliefs about the number of objects in box \(j\) before any searching takes place. Denote by \(\Pi_{nj}\) the probability that box \(j\) contains \(n\) objects, where \(\sum_{n=0}^{\infty} \Pi_{nj} = 1\). It will be assumed that with regard to prior knowledge, distinct boxes may be assumed independent. The value of an object in box \(j\) is \(v_j\) and the cost of
a search of box \( j \) is \( c_j \) where \( v_j > c_j > 0 \). Suppose further that, conditional on the event that there are \( n \) undiscovered objects in box \( j \), the probability that a search of the box will find one of them is \( p_{nj} = 1 - q_{nj} \), where \( q_{nj} \) is monotone nonincreasing in \( n \) for each \( j \). Assume that only one box can be searched at a time and that the \( (t + 1) \)st box to be searched receives a reward which is discounted by a factor \( \alpha^t \). Note that upon discovery, an object is removed from its box. The problem is to find a policy for searching the boxes (or for choosing to search no box at all) which maximises the total expected return.

Benkherouf and Bather [1] proposed the above as a model of a problem in which an oil company has \( N \) areas in which to drill. These areas correspond to the boxes in the above description and the objects to undiscovered sources of oil. Construction of appropriate priors \( \Pi_j \) would make use of relevant geological data. Other correspondences are clear, i.e. \( v_j \) is the value of an oilfield in area \( j \), and so on. Under the constraint that the company cannot (or will not) operate in more than one area at a time what exploration policy maximises the total expected return?

We model this as a stochastic scheduling problem \(( J, 0 \rangle \), where “process job \( j \) ” now becomes “search box \( j \)”. Notice that we have no natural notion of job completion. The state of \( j \) at \( t \) is \( \Pi_j(t) \), the posterior distribution for the number of undiscovered objects in box \( j \). Should a search of box \( j \) be chosen at \( t \), the expected reward is \( \alpha^t R_j(\Pi_j(t)) \) where

\[
R_j(\Pi_j) = v_j \sum_{n=1}^{\infty} \Pi_{nj} p_{nj} - c_j.
\]

Technically, to model the “search no box at all” option we need to include in \( J \) a job with a single state and reward function identically zero. The index for such a job is trivially 0. Note also that the presence of the costs \( c_j \) means that we have departed from the “all rewards nonnegative” assumption. This is easily accommodated here.

Benkherouf et al. [2] approach the problem of computing the index \( G_j(\Pi_j) \) by invoking the notion of an equivalent retirement reward expressed above. In an obvious notation,

\[
G_j(\Pi_j) = \inf \{ M; R_j(\Pi_j, M) = M \}. \tag{2.11}
\]

The decision problem, summarised by data \((\Pi_j, M)\) of choosing between a search of box \( j \) and retirement with reward \( M \) is much simplified by the choice

\[
q_{nj} = (q_j)^n, \quad n \in \mathbb{N} \tag{2.12}
\]

for the detection mechanism. The effect of assumption (2.12) is to make the numbers of successes and failures in past searches of box \( j \) sufficient statistics for the decision problem. Also, for tractable solutions we need to choose \( \Pi_j \) from a conjugate family, i.e. such that posterior distributions also belong to the family. One such choice is \( \Pi_j = E(\lambda_j, q_j) \), the Euler distribution with parameters \( \lambda_j \{e(0, 1) \} \) and \( q_j \) as in (2.12),
This distribution is defined by

\[ \Pi_j = E(\lambda_j, q_j) \Rightarrow \Pi_{nj} = \Pi_{0j} \lambda_j \left\{ \prod_{m=1}^{n} (1 - q_m) \right\}^{-1}, \quad m \in \mathbb{N}. \]

If \( \Pi_j = E(\lambda_j, q_j) \) then it is trivial to show that the posterior distributions following a single success and failure are \( E(\lambda_j, q_j) \) and \( E(\lambda_j q_j, q_j) \) respectively. Hence, the posterior following \( s \) successes and \( f \) failures is \( E(\lambda_j q_f, q_j) \).

The key point to establish about the decision problem summarized by \((\Pi_j, M)\) where \( \Pi_j = E(\lambda_j, q_j) \) is that if retirement is optimal when box \( j \) is in state \( \Pi_j \), then it must also be optimal when box \( j \) is in the state equal to the posterior distribution obtained from \( \Pi_j \) following a single success or failure. With this, it is easy to show from (2.11) that

\[ \Pi_j = E(\lambda_j, q_j) \Rightarrow G_j(\Pi_j) = (-c_j + \lambda_j v_j)(1 - \alpha)^{-1}. \] (2.13)

Here the index may also be thought of as \textit{myopic}, since it is a constant multiple of the one-step reward.

For discussion of other cases, see [2].

The above examples are unusual in yielding job-specific indices of closed form. Usually, some kind of computational approach is needed. The main candidates are described by Katehakis and Veinott [31], who discuss the computational requirements of implementing an index policy.

The alternatives are

(1) value iteration or policy iteration applied to a restart problem, see above;

(2) the "largest remaining index method" due first (I think) to Robinson [38], but developed by Varaiya et al. [40]. This approach computes job-specific indices in a (finite) state space in descending order, with largest first. To summarise, Katehakis and Veinott [31] favour value iteration for a restart problem when transition matrices are sparse (as they often are), the computations are done on line and the state space is large enough. Greatrix [29] is exploring an implementation of this using LP. However, the largest remaining index method is favoured for smaller state spaces.

Following that lengthy discussion of the important special case \( \Gamma = \emptyset \), we now return to general \((J, \Gamma)\). We recall from remark (1) following Definition 4 that when \((J_i, \Gamma_i)\) has a dominating policy \( \hat{\pi}_i \), say, it must follow that whenever an FI policy chooses to process a job from component \( i \) then that job must be chosen according to \( \hat{\pi}_i \), \( 1 \leq i \leq m \). Suppose now that each \((J_i, \Gamma_i)\) has a dominating policy \( \hat{\pi}_i \). Lemma 1 together with a minor elaboration of (a)-(c) above lead to the conclusion that an FI policy now has the following structure:

(i) at each decision epoch \( t \in \mathbb{N} \), select component \( k \) satisfying

\[ G^k(X^k(t)) = \max_{1 \leq j \leq m} G^j(X^j(t)); \]

(ii) from the chosen component \( k \), choose a job according to dominant policy \( \hat{\pi}^k \).
The following key result is due to Whittle [44], who obtained it from his dynamic programing formulation. A more direct proof may be found in Gittins [10]. A converse to Theorem 4 was obtained by Glazebrook [14].

**Theorem 4.** If each \((J_i, F_i)\) has a dominating policy \(\hat{\pi}_i\) then any FI policy is optimal for \((J, F)\) in the class of preemptive (dynamic) policies.

Theorem 5 was originally proved by Glazebrook [12] followed by Meilijson and Weiss [33]. They gave direct arguments. In the current formulation, it follows easily from Theorem 4.

**Theorem 5.** If \(F = \{\text{out-forest}\}\) then every FI policy is optimal for \((J, F)\) in the class of preemptive (dynamic) policies.

In the light of Theorem 4, in order to establish Theorem 5 we need to show that if \(F_i\) is an out-tree, then \((J_i, F_i)\) has a dominating policy. This is trivial if \(|J_i| = 1\). Glazebrook [14] gives a simple induction argument. The nice thing about this out-tree structure for \((J_i, F_i)\) is that at \(t = 0\) \((J_i, F_i)\) has exactly one admissible job \(i\), say. Once \(i\) has been completed the problem of scheduling \(J_i \setminus \{i\}\) still has an out-forest structure. This yields an inductive specification of the dominating policy for \((J_i, F_i)\), namely: process job \(i\) to completion and then operate an FI policy for \(J_i \setminus \{i\}\).

Theorem 5 has proved pivotal in the development of analyses for open problems, i.e. with job (or component) arrivals after time \(t = 0\). See Theorems 3.25 and 3.26 of Gittins [10].

Theorem 5 seems to be as far as one can go on the basis of Theorem 4 in the class of preemptive policies. Gittins [10] gives an example with \(|J| = 3\) and \(F = \{\text{in-tree}\}\) where no dominating policy exists. He also gives a rather difficult condition, grounded in ideas of measurability, which is sufficient for a dominating policy to exist. To the author's knowledge the only use made of this condition to date with regard to preemptive policies has been in the work of Glazebrook [20] and Glazebrook and Gittins [25] concerning the development of performance bounds for FI policies; see Sections 3 and 4.

Suppose now, though, that we extend the ideas of this section to the class of nonpreemptive dynamic (or, equivalently, static list) policies. For example, modify Definition 1 to obtain nonpreemptive Gittins index \(\hat{\alpha}\) by restricting the range of \(\pi\) and \(\tau\), respectively, to nonpreemptive policies and positive-valued stopping times taking values (only) in job completion times. Modify subsequent definitions/results accordingly and derive the class of nonpreemptive forwards induction (NFI) policies by suitable modification of Definition 4. The following result may be deduced from the results of Kadane and Simon [30]. However, the proof they give of their (crucial) Lemma 7 is incomplete. A correct version may be found in Glazebrook and Gittins [24].
Theorem 6. Every NFI policy is optimal for \((J, \Gamma)\) in the class of nonpreemptive (dynamic) policies.

Gittins [10] gives an account of Theorem 6 based on the measurability condition mentioned above. In fact, the problem of determining an optimal static list policy for general \((J, \Gamma)\) is NP-hard. See the Lenstra and Rinnooy Kan [32] discussion of the weighted completion time problem (which may be recovered as the limit of a discounted rewards problem as \(\alpha \to 1\)). However, NFI policies are easy to obtain in special cases; see the Monma and Sidney [34] discussion of \(\Gamma = \{\text{series parallel}\}\). Further, Theorem 6 offers a guide to the development of heuristics in the general case; see [35].

Glazebrook [14] gives conditions on individual jobs in \((J, \Gamma)\), sufficient to ensure the existence of a nonpreemptive policy optimal in the preemptive class. Citing Theorem 6, it will follow that under these conditions, a NFI policy is optimal. To describe these conditions, consider job \(j\) under a policy which processes it nonpreemptively from \(t = 0\) through to completion. Let \(P_j\), the processing requirement, be given by

\[
P_j = \inf \{t; t > 0 \text{ and } X_j(t) \in \Omega_j\}.
\]

We define

\[
m_j(x_j) = E\{x_j^P_j | X_j(0) = x_j\}, \quad x_j \in \Omega_j \setminus \omega_j.
\]

Definition 5. Job \(j\) is shortening if, under any strategy and for any \(t \in \mathbb{N}\),

\[
p [X_j(t + 1) \in \Omega_j \setminus \omega_j, m_j \{X_j(t + 1)\} | X_j(t) = x_j, x_j \in \Omega_j \setminus \omega_j] = 0.
\]

Job \(j\) is improving if, under any strategy and for any \(t \in \mathbb{N}\),

\[
p [X_j(t + 1) \in \Omega_j \setminus \omega_j, G_j \{X_j(t + 1)\} \leq G_j \{X_j(t)\} | X_j(t) = x_j, x_j \in \Omega_j \setminus \omega_j] = 0.
\]

Theorem 7 is proved by Glazebrook [15] using a methodology based around the idea of policy improvement for Markov decision processes.

Theorem 7. If all jobs in \(J\) are both shortening and improving then there is a nonpreemptive policy which is optimal in the class of preemptive policies.

We conclude this section with a short note on single visit policies for \((J, \emptyset)\). Let \(\pi\) be a permutation of \(\{1, 2, \ldots, N\}\) and \(\tau\) a vector of stopping times. Denote by \((\pi, \tau)\) the single visit policy: process \(\pi(1)\) during \([0, \tau_{\pi(1)})\), then \(\pi(2)\) during \([\tau_{\pi(1)}, \tau_{\pi(1)} + \tau_{\pi(2)})\), and so on. We may assume that \(\tau_{\pi(j)}\) depends only upon the evolution of job \(\pi(j)\) and the initial states of \(\pi(i), i \geq j + 1\). Please note that we may have \(\tau_{\pi(j)} = 0\) for some \(j\), i.e. job \(j\) receives no processing. Writing \(R_{\pi(j)} \{\tau_{\pi(j)}, x_{\pi(j)}\}\) for the expected reward obtained from \(j\)'s processing (computed as though this processing began at time 0),
then we express the total expected reward from single visit policy \((\pi, \tau)\) when \(X(0) = x\) as

\[
R_{(\pi, \tau)}(x) = \sum_{j=1}^{N} \left[ \prod_{k=1}^{j-1} E\{x_{\tau(k)}(0) = x_{\pi(k)}\} \right] R_{\pi(j)}(\tau_{\pi(j)}, x_{\pi(j)})\]

The problem of determining optimal \((\pi, \tau)\) jointly is very difficult. However, those of determining the best \(\tau\) for fixed \(\pi\) and of determining the best \(\pi\) for fixed \(\tau\) are both tractable.

**Theorem 8.** (i) If \(\tau_j \neq 0, 1 \leq j \leq N\), then any permutation which orders the jobs according to decreasing values of

\[
R_j(\tau_j, x_j) [E\{1 - \alpha^j \mid X_j(0) = x_j\}]^{-1}
\]

maximises \(R_{(\pi, \tau)}(x)\) over choices of \(\pi\) for fixed \(\tau\);

(ii) for given \(\pi\), any vector \(\tau\) defined by

\[
\tau_{\pi(j)} = \inf\{t; t \geq 0 \text{ and } \gamma_{\pi(j)}(t) < R_{(\pi, \tau)}(x)\},
\]

where

\[
R_{(\pi, \tau)}^{j+1}(x) = \sum_{k=j+1}^{N} \left[ \prod_{i=j+1}^{k-1} E\{x^{\tau(i)} \mid X_{\pi(i)}(0) = x_{\pi(i)}\} \right] R_{\pi(i)}(\tau_{\pi(i)}, x_{\pi(i)})\]

maximises \(R_{(\pi, \tau)}(x)\) over choices of \(\tau\).

The proof of Theorem 8(i) uses a simple pairwise interchange argument while that of Theorem 8(ii) combines use of a backwards induction with an appeal to Theorem 3.

Since for \((\pi^*, \tau^*)\) to be an optimal single visit policy we require *both* that \(\pi^*\) maximise \(R_{(\pi, \tau^*)}(x)\) over choices of \(\pi\) and that \(\tau^*\) maximise \(R_{(\pi^*, \tau^*)}(x)\) over choices of \(\tau\), it is plain that Theorem 8 furnishes us with necessary conditions for the optimality of \((\pi^*, \tau^*)\).

However, in general, these conditions are not sufficient and it seems a far from straightforward matter to establish when they are. Theorem 8 also suggests a natural heuristic approach to the development of good single visit policies. Let \((\pi^0, \tau^0)\) be a single visit policy. Use Theorem 8(i) to determine \(\pi^1\), maximising \(R_{(\pi^0, \tau^0)}(x)\). Now use Theorem 8(ii) to determine \(\tau^1\), maximising \(R_{(\pi^1, \tau^0)}(x)\). plainly, \(R_{(\pi^0, \tau^1)}(x) = R_{(\pi^1, \tau^1)}(x)\). Continue in this fashion until \(\pi^{n+1} = \pi^n\), and then stop. Benkherouf et al. [3] refer to this as the *policy improvement algorithm* (PIA) and they assess its performance computationally on a search problem of the kind described in Example 2. The results are impressive. The single visit policies obtained in this way had expected reward within 1% of the optimum in almost all cases.

### 3. Performance bounds based on reward rates

The idea that the reward rates discussed in Section 2 could be used to develop performance bounds on policies for \((J, \Gamma)\) goes back to Glazebrook [13], who
discussed the properties of a class of policies for multiarmed bandits based on randomised versions of Gittins indices. The ideas in that paper were developed by Glazebrook [16] into a general performance bound for arbitrary policy π for \((J, \emptyset)\). This is given in Theorem 9.

Let \(\pi^*\) be an optimal policy for \((J, \emptyset)\) (see Theorem 3) and let \(G(\pi, x)\) be the Gittins index of the job chosen by \(\pi\) in state \(x\), i.e.

\[
\pi(x) = \text{"process job } j \text{" } \Rightarrow \quad G(\pi, x) = G_j(x).
\]

**Theorem 9 (Performance bound – \((J, \emptyset)\)).**

\[
R_{\pi^*}(x) - R_{\pi}(x) \leq E_x \left( \sum_{t=0}^{\infty} \alpha_t \left[ \max_{1 \leq i \leq N} G_i \{X_i(t)\} - G\{\pi, X(t)\} \right] \right) X(0) = x. \tag{3.1}
\]

Hence, the degree of suboptimality of \(\pi\) is bounded by a natural measure of the extent to which \(\pi\) chooses jobs with smaller Gittins index than does an optimal policy.

Gittins [10] pointed to a weakness in performance of the bound in (3.1) in the limit \(\alpha \to 1\). Work by Katehakis and Veinott [31] and Glazebrook [19, 21] made it possible to develop alternative performance bounds for policies for general \((J, \Gamma)\) with appropriate limiting behaviour. We need the following preliminaries: Denote by \(\{\sigma_n(\pi), n \in \mathbb{N}\}\) the sequence of random times at which policy \(\pi\) switches between jobs, i.e. \(\sigma_0(\pi) = 0\) and, further, \(\pi\) chooses at \(\sigma_n(\pi)\) a job which was not chosen at \(\sigma_{n-1}(\pi)\), \(n \in \mathbb{Z}^+\).

**Definition 6.** The sequence \(\{v_n(\pi), n \in \mathbb{N}\}\) with \(v_0(\pi) = 0\), \(v_{n+1}(\pi) > v_n(\pi), n \in \mathbb{N}\), is at least as fine as \(\{\sigma_n(\pi), n \in \mathbb{N}\}\) if it contains it as a subsequence with probability 1 (w.p.1).

Note that during each interval \([v_n(\pi), v_{n+1}(\pi)]\), a single job (only) is chosen by \(\pi\). Following Glazebrook [10] we develop a reward rate discrepancy measure \(\Delta\) between \(\pi\) and \(\pi^*\) as follows.

**Definition 7.** The discrepancy measure \(\Delta\) defined with respect to sequence \(\{v_n(\pi), n \in \mathbb{N}\}\) at least as fine as \(\{\sigma_n(\pi), n \in \mathbb{N}\}\) is given by

\[
\begin{align*}
(i) & \quad \Delta \{v_n(\pi), v_{n+1}(\pi)\} = 0 \quad \text{if } \pi \text{ is optimal throughout } [v_n(\pi), v_{n+1}(\pi)] \text{ w.p.1. (or if all jobs are complete at } v_n(\pi)); \\
(ii) & \quad \text{otherwise,} \\
\Delta \{v_n(\pi), v_{n+1}(\pi)\} & = G[X\{v_n(\pi)\}] - G_{\pi, v_0(\pi), \ldots, v_n(\pi)}[X\{v_n(\pi)\}].
\end{align*}
\]

Theorem 10 is due to Glazebrook [10].
Theorem 10 (Performance bound \((J, \Gamma)\)).

\[
R_{\pi^*}(x) - R_\pi(x) \leq \mathbb{E}_\pi \left( \sum_{n=0}^{\infty} A \{v_n(\pi), v_{n+1}(\pi)\} \{x^{v_n(\pi)} - x^{v_{n+1}(\pi)}\} \right| X(0) = x.
\]

For \((J, \emptyset)\), the relationship between the bounds in (3.1) and (3.2) is a complex one. Although it seems that Theorem 10 in general offers a more satisfactory approach to evaluating performance than does Theorem 9, it is not true that it dominates it in the sense that there must exist a choice of \(\{v_n(\pi), n \in \mathbb{N}\}\) yielding a value on the r.h.s. of (3.2) smaller than the bound in (3.1). Glazebrook [21] discusses this further and proposes a modification of Theorem 10 for \(\Gamma = \emptyset\) which does achieve that.

Note that it is trivial to deduce from (3.2) that

\[
\{R_{\pi^*}(x) - R_\pi(x)\} \{R_\pi(x)\}^{-1}\]

It has not proved easy to utilise (3.3) for analytical purposes. The main focus has been on the bounds on reward difference \(R_{\pi^*}(x) - R_\pi(x)\) in (3.1) and (3.2). These have occasionally been combined with a lower bound on \(R_\pi(x)\) to yield an upper bound on \(\{R_{\pi^*}(x) - R_\pi(x)\}\{R_\pi(x)\}^{-1}\). One thing which does follow easily from (3.3) is that if for policy \(\pi\) there exists a choice of \(\{v_n(\pi), n \in \mathbb{N}\}\) such that the reward rate earned by \(\pi\) during each interval \([v_n(\pi), v_{n+1}(\pi)]\) comes within \(e\%\) of the Gittins index at \(v_n(\pi)\), i.e.

\[
100 A \{v_n(\pi), v_{n+1}(\pi)\} (G_{v_n(\pi), v_{n+1}(\pi)} [X \{v_n(\pi)\}] - x^{v_n(\pi)})^{-1} \leq e \quad \text{w.p.1.}
\]

then \(R_\pi(x)\) will come within \(e\%\) of \(R_{\pi^*}(x)\).

The bounds in Theorems 9 and 10 have been used for the following purposes.

(a) To evaluate simple heuristics for \((J, \emptyset)\) when the optimal index policies are difficult to construct and/or apply. For example, in the search problem of Section 2 the optimal policy based on the index in (2.13) may involve an unacceptable amount of switching. See [2] for an evaluation of a single visit heuristic based on Theorem 10.

(b) To impose accuracy requirements on numerical procedures for calculating indices which will yield policies to achieve a given level of performance; see [31].

(c) To evaluate FI policies for general \((J, \Gamma)\); see [10].

(d) To develop an approach to sensitivity analysis. Suppose we assume model \((J', \emptyset)\) and compute optimal policy \(\pi^*\) on that basis. However, the true model is \((J, \emptyset)\) which has associated optimal policy \(\pi^*\). An approach to sensitivity analysis espoused
by Glazebrook [18] is to use Theorems 9 and 10 to put an upper bound on $R_{n^*}(x) - R_{n^*}(x)$ (all computations use true model $(J, \emptyset)$) in terms of some natural measure of discrepancy between the true and assumed models.

To give an example of what is possible, suppose that each job in $(J', \emptyset)$ and $(J, \emptyset)$ is a simple job in the sense of Example 1. Suppose also that under both true and assumed models each job has terminal reward $r_j = 1$, but that under assumed model $(J', \emptyset)$ processing time $P_j$ is such that $P_j - 1$ has a Poisson distribution with mean $\lambda_j$, whereas under true model $(J, \emptyset)$, $P_j - 1$ is Poisson with mean $\lambda_j$, $1 \leq j \leq N$. The initial state is that no job has received any processing. An analysis based on a slight development of Theorem 9 yields that

$$|\tilde{\lambda}_j - \lambda'_j| \leq \varepsilon, 1 \leq j \leq N \Rightarrow R_{n^*} - R_{n^*} \leq (1 - \alpha)\varepsilon \{N + \tilde{\lambda}'\}^2 \{1 + \tilde{\lambda}'\}^{-2},$$

where

$$\sum_{j=1}^{N} \tilde{\lambda}_j' \quad \text{and} \quad \tilde{\lambda}' = \min_{1 \leq j \leq N} \lambda_j.$$

Hence, in order that $R_{n^*} - R_{n^*} \leq r$ it is sufficient that all of the assumed parameter values $\lambda'_j$ are within

$$r\{1 + \tilde{\lambda}'\}^2 \{N + \tilde{\lambda}'\}^{-2} (1 - \alpha)^{-1}$$

of their true values.

(e) To develop an approach to modelling issues. This relates closely to (d). We may be considering elaborating a model by including such phenomena as machine breakdowns and/or jobs arrivals. Such elaboration may hugely complicate the search for an optimal policy. However, we can use Theorems 9 and 10 to evaluate heuristics developed for models where these phenomena are excluded as policies for problems where they are present; see [18, 8]. In particular, a “no arrivals” heuristic seems to perform well for open problems much of the time.

A rather different issue which may be dealt with in the same way concerns Bayesian models in which unknown parameters are given prior distributions; see Example 2. Often the search for an optimal policy is simplified if unknown parameters are deemed to be set equal to the mean (say) of the corresponding prior. The question of how well a policy computed on that basis would perform against the (adaptive) optimal policy is related to the concept of “the value of a stochastic solution” (VSS), discussed by Birge [5]. We would hope to use Theorems 9 and 10 to develop bounds in terms of some measure of prior ignorance of the unknown parameters. Glazebrook and Owen [27] are currently developing such an approach.

The bounds (3.1) and (3.2) could be used (but to the author’s knowledge have not been) to make probabilistic statements about the performance of heuristics (i.e. for randomly chosen $(J, \Gamma)$). We now give an example of what might be possible.

Example 3. We shall evaluate a quasi-myopic heuristic $\pi'$ for general $(J, \Gamma)$. This heuristic chooses from among the jobs currently available for processing (call this job
set $A(t)$ at time $t$) one with maximal job-specific Gittins index, i.e.

$$\pi'(X(t)) = \text{"process job } j \text{" } \Rightarrow j \in A(t)$$

and

$$G_j(X_j(t)) = \max_{k \in A(t)} G_k(X_k(t)).$$

We use Theorem 8 to develop a performance bound for $\pi'$ as follows: We may assume that if $\pi'$ chooses to process job $j$ at time $t$, it continues to do so throughout $[t, t + \tau_j(X_j(t))]$ where

$$t + \tau_j(X_j(t)) = \inf\{s; s > t \text{ and } G_j(X_j(s)) < G_j(X_j(t))\}.$$

We now develop sequence $\{v_n(\pi'), n \in \mathbb{N}\}$ as in Definition 6 by requiring $v_0(\pi') = 0$ and

$$\pi'(X(v_n(\pi'))) = \text{"process job } j \text{" } \Rightarrow v_{n+1}(\pi') = v_n(\pi') + \tau_j(X_j(v_n(\pi')))\]$$

Using Theorem 9 and the fact that stopping time $\tau_j(x_j)$ attains the Gittins index $G_j(x_j)$, Glazebrook [10] obtains the inequality

$$R_{\pi^*}(x) - R_{\pi'}(x) \leq E_{\pi'} \left[ \sum_{n=0}^{\infty} \left( \max_{j \in J} G_j(X_j(v_n(\pi'))) - \max_{j \in A(v_n(\pi'))} G_j(X_j(v_n(\pi'))) \right) \times (x^{v_n(\pi')} - x^{v_{n+1}(\pi')}) \right] X(0) = x. \quad (3.4)$$

Suppose now that jobs are chosen independently from some suitable probability space. Denote by $\Gamma$ the initial Gittins index of a job so chosen. We suppose $\Gamma$ to be a random variable with bounded support such that

$$\overline{\Gamma} = \inf\{B; \Gamma \leq B \text{ w.p.1}\} < \infty.$$ In fact, the boundedness of one-step rewards guarantees that $\overline{\Gamma} < \infty$.

We consider a sequence of scheduling problems $\{(J_n, \Gamma_n), n \in \mathbb{N}\}$, with $|J_n|$ and $\Gamma_n$ fixed for a given $n$, but with jobs chosen at random, as above. Use $\phi(n)$ to denote the number of components in $(J_n, \Gamma_n)$. Suppose that $\phi(n) \to \infty$, $n \to \infty$. Use $\phi(n, \delta)$ to denote the set of components with source jobs having job-specific index in the range $[\overline{\Gamma} - \delta, \overline{\Gamma}]$. From (3.4) it is easy to show that

$$R_{\pi^*}^n - R_{\pi'}^n \leq \delta E \left[ 1 - x^{\tau(n, \delta)} \right] + \overline{\Gamma} E \left[ x^{\tau(n, \delta)} \right]. \quad (3.5)$$

In (3.5), $\tau(n, \delta)$ is the first time at which $\pi'$ processes a job outside the set $\phi(n, \delta)$ and superscript $n$ denotes the problem $(J_n, \Gamma_n)$. Please note that both sides of inequality (3.5) are now random variables.
If we now suppose that
\[ p[\Gamma \geq B - \delta] = O(\delta^r) \]
for some \( r > 0 \), then it is easy to conclude from (3.5) that
\[ \beta < r^{-1} \Rightarrow \lim_{n \to \infty} \{ \phi(n) \beta \{ R^n_{\pi^*} - R^n_{\pi^c} \} = 0 \quad \text{w.p.1.} \]

Not surprisingly, the performance of quasi-myopic policy \( \pi^c \) is dependent upon the “shape” of the scheduling problem as reflected in the number of components and upon the proportion of high index jobs. Incidentally, we may easily deduce
\[ \beta < r^{-1} \Rightarrow \lim_{n \to \infty} \{ \phi(n) \beta \{ R^n_{\pi^*} - R^n_{\pi^c} \} \{ R^n_{\pi^c} \}^{-1} = 0 \quad \text{w.p.1.} \]

4. Performance bounds for forwards induction policies

Let \( M \) be a general discounted Markov decision process (MDP) with discount rate \( \alpha \) and bounded rewards, as in Ross [39]. It is plain that all the ideas of Section 2 may be readily adapted to yield FI policies for \( M \). Merely replace \((J, \Gamma)\) by \( M \) throughout. Recall from comment (2) following Definition 4 that the stochastic sequence of Gittins indices \([G\{X(\hat{\delta}_n)\}, n \in \mathbb{N}]\) is decreasing almost surely. Glazebrook and Gittins [25] obtained a performance bound on FI policy \( \pi_G \) for \( M \) in terms of a natural measure of the rate of decrease of this sequence. In Theorem 11, \( \pi^* \) is optimal for \( M \).

**Theorem 11** (Performance bound – FI policies for \( M \)).

\[ R_{\pi^*}(x) - R_{\pi_G}(x) \leq E_{x_0} \left[ \sum_{n=1}^{\infty} \left[ G\{X(\hat{\delta}_{n-1})\} - G\{X(\hat{\delta}_n)\} \right] x^n \right] X(0) = x. \]

Further, Glazebrook and Gittins [25] were able to use Theorem 11 to produce a probabilistic analysis of the performance of FI policies in a complex scheduling environment. Example 4 is a brief summary.

**Example 4.** Let \( M_1, M_2, \ldots, M_n \) be discounted MDPs sharing discount rate \( \alpha \) and each having bounded rewards. From these we construct a new discounted MDP, denoted \( \{(M_1, M_2, \ldots, M_n); m\} \equiv M(n, m) \), where \( m \leq n \), as follows: at each decision epoch \( t \in \mathbb{N} \), we choose a vector of \( m \) actions \((a_{i_1}, a_{i_2}, \ldots, a_{i_m})\) where \( i_r \neq i_s, r \neq s \), and \( a_{i_1} \) is an action for \( M_{i_1} \) in its current state \( X_{i_1}(t) \). When action \((a_{i_1}, a_{i_2}, \ldots, a_{i_m})\) is taken:

(i) the current states of \( M_{i_1}, M_{i_2}, \ldots, M_{i_m} \) all change according to their (individual) transition laws;

(ii) if \( j \neq i_r \) for any \( r, 1 \leq r \leq m \), then the state of \( M_j \) remains unchanged;

(iii) the (discounted) reward earned is the sum of the individual rewards earned by the transitions in \( M_{i_1}, M_{i_2}, \ldots, M_{i_m} \).
If each of the $M_i$'s is simply a job (i.e. with no internal decision structure) then $M(n, m)$ models the kind of multiprocessor scheduling problems considered by Weber [42]. The MDP structure allows us to model multimachine problems with arrivals and/or precedence constraints, etc.

As in Example 3, we suppose that the $M_i$'s are chosen in an independent fashion from some suitable probability space. We again suppose that $\Gamma$, the initial Gittins index of an MDP so chosen, is a random variable such that

$$\bar{\beta} = \inf\{B; \Gamma \leq B \text{ w.p.}1\} < \infty.$$  

We now consider the sequence $\{M(n, m), n - m \in \mathbb{N}\}$ of scheduling problems obtained by choosing MDPs in this way. For given $n$, let $n(\delta)$ be the number of constituent MDPs so chosen whose Gittins index lies in the range $[\bar{\beta} - \delta, \bar{\beta}]$. It can be established from Theorem 11 that (in an obvious notation)

$$R_{n,0}^n - R_{n0}^n = \sum_{i=1}^{n(\delta),m-1} \{ m\delta(1 - x)^{-1}\} \alpha' + m\bar{\beta}x^{[n(\delta),m^{-1}]}(1 - \alpha)^{-1}. \quad (4.1)$$

If we now suppose that

$$p(\Gamma \geq \bar{\beta} - \delta) = O(\delta^r)$$

for some $r > 0$, then by appeal to (4.1) and standard results we have that

$$\beta < r^{-1} \Rightarrow \lim_{n \to \infty} n^\beta \{ R_{n,0}^n - R_{n0}^n \} = 0 \quad \text{w.p.}1.$$  

5. Further study

There are many aspects of the ideas presented in this paper which remain unclear. Suggestions for further study are the following.

(i) We need further elucidation of when the FI approach is an attractive alternative to conventional dynamic programming. This concerns both the scope of any computational savings achieved and the performance of the policies so constructed. Notwithstanding the work of Glazebrook [20], and Glazebrook and Gittins [25] there is still relatively little we can say with confidence about the performance of FI policies for general $(J, F)$. Some numerical work would be helpful.

(ii) In many contexts it may be worthwhile exploring the properties of heuristics which have an FI element while not necessarily being fully FI policies. One example is the class of cost-rate heuristics for semi-Markov decision processes proposed by Glazebrook et al. [23].

(iii) The assumption that jobs not being processed remain frozen is fundamental to the results in the paper. Authors, including Whittle [46], Weber and Weiss [43] and Glazebrook and Whitaker [28] have sought to relax that condition in various ways. The latter paper concerns scheduling models where there are dependencies between the jobs. This is a major area about which very little is known. As Bergman and
Gittins [4] point out, such models are important for applications in the area of drug screening.

(v) The idea of probabilistic analysis for heuristics in stochastic scheduling is a relatively new one. There is great scope here – particularly with regard to complex scheduling models, as illustrated in Section 4.

(vi) Many of the ideas of Section 2 carry over to the generalised bandits of Nash [36], with their multiplicatively separate reward structure; see [22] for a recent account. The indices encountered in such problems can be negative. When such negative indices are encountered, there is much about the behaviour and structure of FI policies which is unclear.

(vii) It would be helpful to know whether there are simply stated conditions under which the PIA described at the end of Section 2 yields an optimal single visit policy.

References