Adv. Appl. Prob. **26,** 474–479 (1994)
Printed in N. Ireland

OPTIMAL STOCHASTIC SCHEDULING OF FOREST NETWORKS WITH SWITCHING PENALTIES

MARK P. VAN OYEN AND
DEMOSTHENIS TENEKETZIS,* *University of Michigan, Ann Arbor*

Abstract

We present structural properties of optimal policies for the problem of scheduling a single server in a forest network of N queues (without arrivals) subject to switching penalties. In addition to linear holding costs, we impose either lump sum switching costs or batch set-up delays which are incurred at each instant the server processes a job in a queue different from the previous one. We use reward rate notions to unearth conditions on the holding costs and service distributions for which an exhaustive policy is optimal. For the case of two nodes connected probabilistically in tandem, we explicitly define an optimal policy under similar conditions.

OPTIMAL CONTROL OF QUEUES; QUEUEING NETWORKS; SWITCHING COST; SWITCHING TIME; MULTI-ARMED BANDITS; COUPLING

AMS 1991 SUBJECT CLASSIFICATION: PRIMARY 60K25
SECONDARY 90B35

1. Introduction

Applications such as computer networks, data networks, and manufacturing systems have motivated the efforts of researchers to address scheduling subject to switching penalties (see Browne and Yechiali (1989); Bruno and Downey (1978); Glazebrook (1980); Gupta et al. (1987); Hofri and Ross (1987); Liu et al. (1992); Magnanti and Vachani (1990); Monma and Potts (1989); Perkins and Kumar (1989); Rajan and Agrawal (1991); Santos and Magazine (1985); Van Oyen (1992); and Van Oyen et al. (1992)). Although it is often realistic to include a penalty for each switch from one project type to another, few results are known regarding optimal policies for such deterministic scheduling problems and very few for such stochastic problems. See Glazebrook (1980); Gupta et al. (1987); Hofri and Ross (1987); Liu et al. (1992); Rajan and Agrawal (1991); Van Oyen (1992); and Van Oyen et al. (1992) for the *stochastic* scheduling literature treating switching penalties.

Monma and Potts (1989) analyze a variety of *deterministic* scheduling models that include batch set-up times (delays) under the optimization criteria of total weighted

This work was supported in part by a Department of Electrical Engineering and Computer Science Graduate Fellowship and by NSF grant No. NCR-9204419.

Received 27 July 1992; revision received 11 June 1993.

* Postal address: Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI 48109-2122, USA.

completion time, maximum completion time, maximum lateness, and number of late jobs. For certain problems, optimal policies are partially characterized and computational algorithms based on dynamic programming are developed. Because of the computational difficulty of these algorithms, Monma and Potts suggest that the development of heuristics for scheduling with set-up times (delays) is a topic of practical importance. We concur with their views, and toward this end we seek to rigorously develop qualitative results for stochastic scheduling with switching penalties, thereby providing insight into the character of such problems and a foundation for improved design and heuristics.

Rajan and Agrawal (1991) study the stochastic scheduling of a single server in a system with switching costs (or switchover times) and a general exogenous arrival process which is uniformly split amongst p queues. The queues are identical with respect to service period, holding cost, and switching penalty; thus, the system possesses complete symmetry. They define optimality in terms of a stochastic dominance of the cost process under one policy over any other policy. Under a restriction to non-idling policies, they show that it is optimal to serve exhaustively and to allocate the server to the longest available queue at each epoch of exhaustion. The effects of initial customer configuration are studied and extensions to switchover times and to partial information are treated. Liu et al. (1992) study similar problems with switching times under the objectives of stochastically minimizing either the total unfinished work in the system or the total number of jobs in the system. They devote considerable effort to the determination of conditions under which optimal policies can be characterized as being greedy, exhaustive, non-idling, patient, or impatient. Moreover, they characterize optimal policies under a variety of information patterns: complete, partial, periodic, delayed, or nearest neighbor. We note, however, that the majority of their analysis requires the assumption of complete symmetry in the system.

A treatment of connected, heterogeneous queues subject to either switching delays or switching costs is the primary novelty of our contribution. To the best of our knowledge, the existing literature does not address the issues introduced when heterogeneous queues are connected together. In Van Oyen et al. (1992), we analyzed a problem of parallel, heterogeneous queues with no arrivals and demonstrated the optimality of an index policy. Crucial to that result was the optimality of an exhaustive policy. Since the present formulation allows jobs served at one queue to be transferred to another, the system at hand may be said to possess *internal* arrivals. This addition introduces a substantial difficulty and destroys the simple index structure found in the case of parallel queues. If each queue is viewed as an arm of a multi-armed bandit, then the connections cause the bandit to be *restless* in the terminology of Whittle (see Whittle (1988) and Weber and Weiss (1990)). Regardless of the perspective taken, connections complicate the nature of an optimal policy. We are able, however, to delineate a class of problems for which relatively simple policies (exhaustive ones) are optimal.

Systems of connected queues for which switching is not penalized have enjoyed extensive treatment. Klimov (1974), (1978) identified the optimality of a strict priority rule for general topologies of interconnected queues with Poisson arrivals, linear holding costs, and no switching penalties. See also Foss (1984); Harrison (1975); Lai and Ying (1988); Nain (1989); and Nain et al. (1989).

The paper is organized as follows. We formulate the switching cost and switching delay problems in Section 2, then analyze them in Section 3. Section 3.1 introduces basic reward and reward rate notions as preliminaries. A class of problems with deterministic connections for which exhaustive policies are optimal is identified in Section 3.2. In Section 3.3 we explicitly define an optimal policy for two queues connected stochastically in tandem. Finally, Section 3.4 addresses some issues pertaining to the search for an optimal solution to the class of problems identified in Section 3.2.

2. Problem formulation

A single server is to be allocated to jobs in a system of queues with no exogenous arrivals. The system contains N queues which are connected in the sense that a job completed in queue n ($n \in \{1, 2, \cdots, N\}$) either leaves the system with probability one or re-enters the system at queue $I(n)$ with probability 1, where $I(n) \in \{1, \cdots, n-1, n+1, \cdots, N+1\}$ with $I(n) = N+1$ denoting the case where jobs of node n exit the system. We assume that no job can visit any queue more than once, and hence every job eventually leaves the system. Thus, the directed graph associated with the queueing network can be described as a forest, that is, a collection of one or more trees. Each queue, n, possesses a general (positive) service period distribution with mean μ_n^{-1} such that $0 < \mu_n^{-1} < \infty$. Successive services in node n are independent and identically distributed (i.i.d.) and independent of all else.

Holding cost is assessed at a rate of c_n ($c_n \geqq 0$) cost units per job per unit time spent in queue n (including time in service). A set-up cost K_n ($K_n > 0$) is incurred at each instant (including time 0) the server processes a job in a queue n different from that of the previous job. With $\mathbb{R}^+(\mathbb{Z}^+)$ denoting the non-negative reals (integers), let $\{X_n^g(t): t \in \mathbb{R}^+\}$ be the right-continuous queue length process of node n under policy g (including any customer of node n in service). Denote the vector of initial queue lengths by $x = (x_1, x_2, \cdots, x_N) = X(0^-) \in (\mathbb{Z}^+)^N$, where x is fixed. The policy g specifies, at each decision instant, the queue to be served or to be set up. Without loss of generality, the jobs within a given queue are assumed to be served according to an arbitrary ordering. Let $n^g(t)$ be the right-continuous process describing the location of the server at t under policy g. Define $\Gamma_n^g = \{t \in \mathbb{R}^+: n^g(t^-) \neq n, n^g(t) = n\}$ to be the set of random instances of switching into node n under g. For the fixed

initial state $X(0^-) = x$, the total expected α-discounted cost of policy g, $J(g)$, can now be expressed as

$$J(g) = \boldsymbol{E}\left\{\int_0^\infty e^{-\alpha t}\left(\sum_{n=1}^{N} c_n X_n^g(t)\right) dt + \sum_{n=1}^{N}\sum_{t\in\Gamma_n^g} K_n e^{-\alpha t}\right\}, \tag{2.1}$$

where $\alpha \geqq 0$. Note that our notation $J(g)$ does not explicitly indicate the dependence of (2.1) on the initial state x, because x is fixed and also because this simpler notation suits our purposes. The objective is to determine a policy $g^* \in G$ that minimizes $J(g)$, where G, the class of admissible strategies, is taken to be the set of non-idling, non-preemptive, and non-anticipative policies. Our restriction to non-preemptive policies implies that once the service of a job begins, that service cannot be discontinued, nor can it be interrupted by switching. Switches occur only at job completion epochs. Without loss of optimality, the minimization problem can be restricted to the class of pure Markov policies (see Ross (1983)). Since jobs are routed deterministically in the network, the class of pure Markov policies is equivalent to G^L, the class of list policies. Thus $g \in G^L$ can be regarded as a string $g_1 g_2 \cdots g_k$ which specifies that the lth job is served in node $g_l \in \{1, 2, \cdots, N\}$ and k is used here to denote the total number of jobs to be served. In what follows, we use the term *feasible* to describe any policy g that specifies a service ordering that is consistent with the topology and initial state of the network.

Although our presentation emphasizes the case of switching *cost*, we also address the case of switching *delay* which we deem to be at least as important. The switching (set-up) delay, D_n, represents a period of time which is required to prepare the server for processing a job of queue n, if n differs from the previous queue served. We assume that successive set-ups for node n require positive, i.i.d., finite mean delays independent of all else. The restriction to non-preemptive policies is understood to disallow reneging during set-up periods. Such a problem (with $K_n = 0$ for all n) will be referred to as the switching delay problem. The objective is to minimize over $g \in G^L$

$$\bar{J}(g) = \boldsymbol{E}\left\{\int_0^\infty e^{-\alpha t}\left(\sum_{n=1}^{N} c_n X_n^g(t)\right) dt\right\}. \tag{2.2}$$

The remainder of the formulation is as before.

3. Analysis

We use the following terminology. For $i, n \in \{1, 2, \ldots, N\}$ we say that a node n is *upstream* of node i if $I(n) = i$ or if there exist intermediate nodes $j_1, j_2, \cdots, j_m \in \{1, 2, \cdots, N\}$ such that $I(n) = j_1$, $I(j_1) = j_2, \cdots, I(j_m) = i$. Analogously, node n is *downstream* of i if the network allows jobs to proceed from i to n. We call a job in node n a *descendant* of a job in node i if the job in n was not

originally in n, but resulted (perhaps indirectly) from serving a job in node i. Thus a job in mode n will eventually generate one descendant in each node downstream of n.

If switching is not penalized (i.e. $K_n = 0$ and $D_n = 0$), the problem of Section 2 possesses a solution described by an index rule and can be solved in a number of ways. One way is via the theory of priority queues initially developed by Klimov (1974), (1978) (see also Harrison (1975); Nain et al. (1989); Varaiya et al. (1985); and Walrand (1988)). Regardless of the details of the technique employed in the aforementioned references, the result is that an index can be associated with each job in the system. It is optimal to serve, at each instant, the job possessing the greatest index. The index is a measure of the reward rate that can be achieved by serving a job and is defined in Nain et al. (1989) or Varaiya et al. (1985). Because all jobs in a given queue are indistinguishable, the index can be associated with the queue, rather than with individual jobs. When switching penalties are considered, the switching penalty requires one to consider a group of jobs rather than an individual job. The dependence of reward rates on the particular queue lengths greatly complicates the problem.

Another way to view the problem without switching penalties is via the theory of scheduling subject to precedence constraints. Section 3.10 of Gittins (1989) exposits the scheduling of jobs subject to precedence constraints. The relationship of our problem to Gittins' formulation of scheduling subject to precedence constraints can be seen as follows. Consider the set, $\mathscr{X}$, of all jobs that will ever be served in the system; that is, those initially present and their descendants. If suffices to identify each job in $\mathscr{X}$ with a distinct queue and an associated precedence constraint. That is, if the job x' is an immediate descendant of a job x and x' in turn possess an immediate descendant x'', then precedence constraints are needed to require that the processing of x precedes x' and similarly x' precedes x''. Thus for every job originally in the system, the reformulated problem contains m $(1 \leqq m \leqq N)$ queues, each possessing a single job having either no precedence constraint (if initially present) or one precedence constraint (if a descendant), as dictated by the connection structure. Thus Corollary 3.21 of Gittins treats the case without switching penalties. As we have previously noted, switching costs (or delays) cannot be associated with an individual job; rather, they must be tied to a class or group of jobs. Thus one is forced to abandon the simple model of precedence constraints and fixed costs/rewards associated with each job.

The preceding discussion provides a perspective on the difficulties introduced by switching penalties. As we shall show, index rules like Klimov's (1974), (1978) are no longer optimal in general. To proceed with our analysis, we first introduce notions of reward and reward rate that will be used to prove the main result.

3.1. *Reward and reward rate notions.* In this section we begin with the discounted problem with switching cost. The case $\alpha = 0$ is treated later as a limiting case. We extract from the cost criterion (2.1) expressions for the reward associated

with a particular sequence of actions under the assumption $\alpha > 0$. These rewards in turn define associated reward rates. Together, they prove crucial to the development of qualitative and quantitative properties of optimal policies. At the end of this section, we treat the problem with switching delay.

Let $Y_l^g(t)$ denote the cumulative number of departures from node l through time t under g. Thus,

$$X_n^g(t) = x_n + \sum_{i:I(i)=n} Y_i^g(t) - Y_n^g(t), \tag{3.1}$$

where the sum $\Sigma_{i:I(i)=n} Y_i^g(t)$ is vacuous if there are no nodes upstream of n. Using (3.1), the cost criterion (2.1) with $X(0^-) = x$ can be expressed as

$$J(g) = \sum_{n=1}^{N} c_n x_n \alpha^{-1} - \boldsymbol{E}\left\{ \sum_{n=1}^{N} \int_0^\infty c_n \left(Y_n^g(t) - \sum_{i:I(i)=n} Y_i^g(t) \right) e^{-\alpha t}\, dt - \sum_{n=1}^{N} \sum_{t \in \Gamma_n^g} K_n e^{-\alpha t} \right\}. \tag{3.2}$$

Equation (3.2) demonstrates that minimization of the original *cost* $J(g)$ is equivalent to maximization of the expected discounted *reward* $R(g)$, where

$$R(g) = \boldsymbol{E}\left\{ -\sum_{n=1}^{N} \sum_{t \in \Gamma_n^g} K_n e^{-\alpha t} + \sum_{n=1}^{N} \int_0^\infty c_n \left(Y_n^g(t) - \sum_{i:I(i)=n} Y_i^g(t) \right) e^{-\alpha t}\, dt \right\}. \tag{3.3}$$

To interpret $R(g)$, note that the cost function $J(g)$ is composed of three constituent elements: (1) the infinite-horizon holding cost incurred if one never serves the jobs initially in the system, (2) the holding cost that is saved by service under g, and (3) the switching penalties paid to achieve service under g. Since the first element is constant for all policies, the reward of policy g, $R(g)$, can be interpreted as the holding cost saved under g minus the switching penalties incurred under g.

Consider a policy π that sets up node j at time $t = 0$, serves u ($u \leqq x_j$) jobs, and idles thereafter. Policy π is defined by the string $\pi_1 \pi_2 \cdots \pi_u$ with $\pi_i = j$ for all i, where π_i denotes the queue in which the ith job is served. For our purposes, it is useful to compute the reward of this sequence of actions. Let $f_{j,i}$ denote the processing time required to complete i jobs in node j, and let $S_j \triangleq \boldsymbol{E}\{\exp(-\alpha f_{j,1})\}$ be the expected discounted service period for a single job in node j. Since successive service periods in j are i.i.d., $\boldsymbol{E}\{\exp(-\alpha f_{j,i})\} = S_j^i$ and we find

$$\begin{aligned} R(\pi) &= -K_j + \boldsymbol{E}\left\{ \sum_{i=1}^{u} \left[\int_{f_{j,i}}^{\infty} c_j e^{-\alpha t}\, dt - \int_{f_{j,i}}^{\infty} c_{I(j)} e^{-\alpha t}\, dt \right] \right\} \\ &= -K_j + \sum_{i=1}^{u} S_j^i (c_j - c_{I(j)}) \alpha^{-1} \\ &= -K_j + (c_j - c_{I(j)}) S_j (1 - S_j)^{-1} \alpha^{-1} (1 - S_j^u), \end{aligned} \tag{3.4}$$

where $c_{N+1} \triangleq 0$. Note that provided $u \leqq x_j$, $R(\pi)$ does not depend on the system queue lengths at time 0.

For $n = 1, 2, \cdots, N$, let

$$h_n \triangleq (c_n - c_{l(n)})S_n(1 - S_n)^{-1}, \qquad \alpha > 0 \tag{3.5}$$

where h_n can be interpreted as the (equivalent constant) rate at which holding cost is reduced by working in node n. To see this, notice that $\alpha^{-1}(1 - S_j^u) = \boldsymbol{E}\{\int_0^{f_{j,u}} e^{-\alpha t}\, dt\}$ in (3.4). For a given string g, let $\bar{\nu}(g)$ denote the expected discounted reward earned per expected unit of discounted time (hereafter referred to simply as *reward rate*) associated with policy g. Similarly, let $\nu_j(u)$ denote the reward rate associated with the service of $u \in \mathbb{Z}^+$ jobs in node $j \in \{1, 2, \cdots, N\}$. Then for $\alpha > 0$ and policy π as previously defined, (3.4) yields

$$\bar{\nu}(\pi) = \nu_j(u) = R(\pi)/\boldsymbol{E}\left\{\int_0^{f_{j,u}} e^{-\alpha t}\, dt\right\} \tag{3.6}$$

$$= h_j - \alpha K_j/(1 - S_j^u). \tag{3.7}$$

Thus, reward can in turn be easily described in terms of reward rate for problems with discounting:

$$\begin{aligned} R(\pi) &= \bar{\nu}(\pi)\boldsymbol{E}\left\{\int_0^{f_{j,u}} e^{-\alpha t}\, dt\right\} \\ &= \nu_j(u)\alpha^{-1}(1 - S_j^u). \end{aligned} \tag{3.8}$$

Equations (3.7) and (3.8) are the reward rate and reward associated with a 'block' of consecutive jobs of the same type. They form the basis for evaluating more general policies. If $g \in G^L$, then g can be written in *block form* as

$$g = j(1)^{u(1)} \| j(2)^{u(2)} \| \cdots \| j(q)^{u(q)} = B_1 \| B_2 \| \cdots \| B_q, \tag{3.9}$$

where $j(i)^{u(i)}$ denotes a string of $u(i)$ jobs of type $j(i)$; $\|$ denotes string concatenation; $u(i)$, $q \in \mathbb{N}$ (the set of natural numbers); $j(i) \neq j(i+1)$; and $B_i = j(i)^{u(i)}$. $R(g)$, the reward of string g, can be computed as follows. Let $\tau_i = f_{j(i),u(i)}$, the service period associated with block B_i; $s_i = \boldsymbol{E}\{\exp(-\alpha\tau_i)\}$; and $b(i) = \sum_{l=1}^{i-1} \tau_l$. Then

$$\begin{aligned} R(g) &= \boldsymbol{E}\left\{\sum_{i=1}^{q} e^{-\alpha b(i)} R(B_i)\right\} \\ &= \sum_{i=1}^{q} \left(\prod_{l=1}^{i-1} s_l\right) R(B_i). \end{aligned} \tag{3.10}$$

Using (3.8) and (3.10), the reward rate of g can be decomposed and written in terms

of constituent reward rates as

$$\begin{aligned}\bar{v}(g) &= \boldsymbol{E}\left\{\sum_{i=1}^{q}\left(\prod_{l=1}^{i-1} s_l\right)R(B_i)\right\}\Big/\boldsymbol{E}\left\{\int_0^{b(q+1)} e^{-\alpha t}\,dt\right\} \\ &= \left[\sum_{i=1}^{q}\left(\prod_{l=1}^{i-1} s_l\right)\bar{v}(B_i)(1-s_i)\right]\left[1-\prod_{i=1}^{q} s_i\right]^{-1},\end{aligned} \tag{3.11}$$

where $\alpha > 0$, $s_l = S_{j(l)}^{u(l)}$ and $\bar{v}(B_i) = v_{j(i)}(u(i))$. As will be seen in the proofs of our results, the reward rates, $\bar{v}(\cdot)$, play a key role in comparing the rewards, $R(\cdot)$, earned under alternative policies (which will be constructed as local perturbations of each other).

Remark 1. For the problem with $\alpha = 0$, we define h_n to be the limit, as $\alpha \searrow 0$, of the product of α and (3.5):

$$h_n \triangleq (c_n - c_{I(n)})\mu_n, \qquad \alpha = 0. \tag{3.12}$$

Remark 2. In the problem with switching delays, the rewards and reward rates differ slightly. We point out only the differences that remain after setting $K_n = 0$ for all n. We use the notation previously developed. Instead of (3.4), the reward for serving u jobs in node j is, for $\alpha > 0$,

$$\begin{aligned}R(\pi) = R(j^u) &= \boldsymbol{E}\left\{\sum_{i=1}^{u}\left[\int_{D_j+f_{j,i}}^{\infty}(c_j - c_{I(j)})e^{-\alpha t}\,dt\right]\right\} \\ &= \boldsymbol{E}\{\exp(-\alpha D_j)\}(c_j - c_{I(j)})S_j(1-S_j)^{-1}\alpha^{-1}(1-S_j^u).\end{aligned} \tag{3.13}$$

The definition of h_n remains unchanged. The reward rate of (3.7) becomes

$$\bar{v}(j^u) = v_j(u) = R(\pi)/\boldsymbol{E}\left\{\int_0^{D_j+f_{j,u}} e^{-\alpha t}\,dt\right\} \tag{3.14}$$

$$= \boldsymbol{E}\{\exp(-\alpha D_j)\}h_j(1-S_j^u)/(1-S_j^u\boldsymbol{E}\{\exp(-\alpha D_j)\}). \tag{3.15}$$

Thus, (3.8) is now

$$R(j^u) = v_j(u)\alpha^{-1}(1-S_j^u\boldsymbol{E}\{\exp(-\alpha D_j)\}). \tag{3.16}$$

The remainder of the development applies, provided we redefine the following quantities. Let the time required to set up and to serve the ith stage of policy g (see (3.10)) be defined as $\tau_i = D_{j(i)} + f_{j(i),u(i)}$. Thus,

$$s_i = \boldsymbol{E}\{\exp(-\alpha\tau_i)\} = \boldsymbol{E}\{\exp(-\alpha D_{j(i)})\}S_{j(i)}^{u(i)}.$$

In Section 3.4, we present an investigation of approaches for simplifying the search for an optimal policy. So as to make that discussion precise in the context of switching delays and no discounting, we present here the necessary reward rate

quantities. Recall that (3.12) defines h_n for undiscounted problems. For a block of u jobs in node j, we get $\bar{v}(j^u)$ as the limit as $\alpha \searrow 0$ of (3.15). Thus,

$$\bar{v}(j^u) = v_j(u) = h_j \frac{u\mu_j^{-1}}{u\mu_j^{-1} + \boldsymbol{E}\{D_j\}}, \qquad \alpha = 0. \tag{3.17}$$

Similarly, the limit as $\alpha \searrow 0$ of (3.13) times α yields

$$R(j^u) = h_j u \mu_j^{-1} = u(c_j - c_{I(j)}), \qquad \alpha = 0. \tag{3.18}$$

For a policy g as in (3.9), we get a reward rate of

$$\bar{v}(g) = \sum_{i=1}^{q} R(j(i)^{u(i)}) \Big/ \sum_{i=1}^{q} (u(i)\mu_{j(i)}^{-1} + \boldsymbol{E}\{D_{j(i)}\}), \qquad \alpha = 0. \tag{3.19}$$

3.2. *The main result.* The presence of node connections creates a problem considerably more complicated than the scheduling of parallel queues with switching cost, for which an index policy is optimal (see Van Oyen et al. (1992)). For the problem of connected queues formulated in Section 2, we believe that the incentive to switch to a particular queue is increasing in its queue length. Thus, we conjecture that a threshold-type policy is optimal in general (that is, for each queue there is a switching curve that depends on all other queue lengths). We do not prove this conjecture; instead, we use the reward rate notions developed in Section 3.1 to prove for certain problems the optimality of *exhaustive* service, as defined below.

Definition 1. A policy g is said to be *exhaustive* if according to g the server switches out of node n at decision instant t only if queue n is empty at t.

Note that an exhaustive policy may visit a node multiple times because of the connection structure. We now derive the main result of the paper.

Theorem 1. If the connected queues can be labeled such that

$$h_1 \geqq h_2 \geqq \cdots \geqq h_N \quad \text{and} \quad I(n) > n \qquad \text{for } n = 1, 2, \cdots, N, \tag{3.20}$$

then only an exhaustive policy can be optimal for the stochastic scheduling problem with a cost criterion given by (2.1). Otherwise, an exhaustive policy is not in general optimal.

Discussion. The results of Theorem 1 might be argued for on intuitive grounds using the following reasoning:

Suppose first that the queues are labeled such that $h_1 \geqq h_2 \geqq \cdots \geqq h_N$ and $I(n) > n$ for $n = 1, 2, \cdots N$. Assume that at time $t = 0$ it is optimal to serve node j; let $y < x_j$ jobs be served consecutively in node j during $[0, t_s)$ where t_s is a decision epoch. We

consider whether there is any incentive to switch from node j at t_s. At time t_s the reward rate for serving one or more jobs in node j is h_j, since the switching cost K_j was already paid at time 0 upon set-up. There ought not to be any incentive to switch to any nodes upstream of j, since it was orignally preferable to serve node j, and the reward rate of any node upstream of j remains constant (the number of customers in these nodes is not affected by the service of node j). Similarly, there is no incentive to switch to any node k such that $I(j) \neq k$. It remains to consider node $I(j)$ whose queue length has grown from $x_{I(j)}$ at time 0 to $x_{I(j)} + y$ at time t_s. Since $h_j \geqq h_{I(j)} > v_{I(j)}(l)$ for any number of jobs l, the reward rate of node $I(j)$ cannot surpass that of continuation in node j. Thus, it is intuitively plausible, under the above assumptions, that once a node is set up, it should be cleared before switching to another. This is rigorously shown in the proof of the theorem.

Suppose next that $h_1 \geqq h_2 \geqq \cdots \geqq h_N$, $I(n) > n$, $n = 1, 2, \cdots N$, is *not* true. Assume at time 0 it is optimal to serve node j and $h_{I(j)} > h_j$. As before, let $y < x_j$ jobs be served consecutively in node j during $[0, t_s)$. At time t_s the reward rate for serving one or more jobs in node j is h_j; the reward rate of serving node $I(j)$ is $v_{I(j)}(x_{I(j)} + y)$. It is possible, for certain $x_{I(j)}$ and y, to have $v_{I(j)}(x_{I(j)} + y) > h_j$ since $h_{I(j)} > h_j$ and $v_{I(j)}(l) \nearrow h_{I(j)}$ as $l \to \infty$. Consequently, at t_s it may be optimal to switch from node j to node $I(j)$. This is illustrated by an example at the conclusion of the proof of the theorem.

Note that the statement of Theorem 1 does not provide any information about the optimal order of scheduling (i.e. optimal exhaustive list policies). Determining an optimal exhaustive list policy remains a difficult unsolved problem. In Section 3.4 we indicate how the notions of reward rate can be used to simplify the search for an optimal policy.

In the case that $c_n = 0$ for all n, the problem reduces to that of minimizing the cost due to switching and the result implies that for any forest, an optimal policy is exhaustive.

Before detailing the proof of Theorem 1, we present three lemmas that develop basic properties of rewards and reward rates. In the following lemmas, M_0, M_1, M_2, and M_3 denote (partial) job lists such that the final job of M_i is of a different queue than the first job of M_{i+1}. Let τ_i denote the time required to process M_i and $s_i = \boldsymbol{E}\{\exp(-\alpha\tau_i)\}$. The first lemma follows directly from (3.11) and Corollary 1 follows in turn from it and (3.7).

Lemma 1. $\bar{v}(M_1 \parallel M_2) \leqq \max\{\bar{v}(M_1), \bar{v}(M_2)\}$.

Corollary 1. For a network satisfying $h_1 \geqq h_2 \geqq \cdots \geqq h_N$, if a job list M_1 contains only jobs in queues $n, n+1, \cdots, N$, then $\bar{v}(M_1) < h_n$.

Lemma 2. Suppose $\bar{v}(M_1 \| M_2 \| M_3) > r$, $\bar{v}(M_3) > r$, and $\bar{v}(M_2) \leqq r$ for some $r \in \mathbb{R}$; then $\bar{v}(M_1 \parallel M_3) > r$.

Proof. Assume that a switch is required between M_1 and M_3 to process $M_1 \parallel M_3$;

otherwise, the result follows *a fortiori* since $M_1 \| M_3$ requires one less switch than our calculations include. Using (3.11) and $s_i < 1$, $\bar{v}(M_1 \| M_2 \| M_3) > r$ implies

$$\begin{aligned} 0 &< (\bar{v}(M_1) - r)(1 - s_1) + s_1(\bar{v}(M_2) - r)(1 - s_2) \\ &\quad + s_1 s_2(\bar{v}(M_3) - r)(1 - s_3) \\ &\leqq (\bar{v}(M_1) - r)(1 - s_1) + s_1 s_2(\bar{v}(M_3) - r)(1 - s_3) \\ &< (\bar{v}(M_1) - r)(1 - s_1) + s_1(\bar{v}(M_3) - r)(1 - s_3), \end{aligned}$$

from which we conclude the result using (3.11) again.

Lemma 3. Let $g \triangleq M_0 \| j^u \| M_2 \| M_3$, where $M_1 = j^u$ and M_2 contains no descendants of j^u. Then $\tilde{g} \triangleq M_0 \| M_2 \| j^u \| M_3$ is a feasible list policy. If $\bar{v}(j^u) < \bar{v}(M_2)$, then $R(\tilde{g}) > R(g)$.

Proof. Assume the interchange of j^u and M_2 does not result in fewer switches for $\tilde{g}$; otherwise, the result holds *a fortiori*. Using (3.10) and (3.11), we get:

$$\begin{aligned} R(\tilde{g}) - R(g) &= \alpha^{-1} s_0[\bar{v}(M_2)(1 - s_2) + s_2 \bar{v}(j^u)(1 - s_1) \\ &\quad - \bar{v}(j^u)(1 - s_1) - s_1 \bar{v}(M_2)(1 - s_2)] \\ &= \alpha^{-1} s_0(1 - s_1)(1 - s_2)[\bar{v}(M_2) - \bar{v}(j^u)] \\ &> 0. \end{aligned}$$

We are now prepared to prove the main result.

Proof of Theorem 1. Suppose $g \in G^L$ (a list policy) is not exhaustive. Since g is not exhaustive, there exists some node n and time t_s at which g switches from queue n to another while jobs remain in n. List g can be written as $M' \| n^y \| M''$, where y is the number of jobs in n served prior to t_s. Without loss of generality, let $M' = \varnothing$ (where $\varnothing$ denotes the null string); thus $g = n^y \| M''$ with $y < x_n$. Since all jobs in a given node are indistinguishable, we may assume without loss of generality that within a queue the jobs are processed first-come-first-served. We develop M'' more explicitly by considering the second visit of g to n: two scenarios are possible. In the first, during its second visit to n, g completes the x_n jobs initially in node n plus some number $z' \in \mathbb{Z}^+$ of jobs which are descendants of jobs served between the first and second visits to n. With $z \triangleq x_n - y$, exactly $z + z'$ jobs are served on the second visit to n. In the second scenario, the total number of jobs served in the first two visits to n is less than x_n. For this case, let $z < x_n - y$ denote the number served in n on the second visit. Thus, in either scenario, it suffices to consider g of the general form $g = n^y \| M'' = n^y \| M \| n^{z+z'} \| M'''$, where $z \leqq x_n - y$ and z' is zero if $z < x_n - y$.

We proceed now to modify g and construct an exhaustive policy that strictly improves g. For technical reasons, the modification of g proceeds according to two cases. We present first the main idea behind each case, then we provide the technical

details of the proof. Case I constructs a right-modification that moves z jobs of node n, (n^z), from the second visit of node n to the first visit of the server to node n (and thereby shifts M to the right). Case II constructs a left-modification which serves the y jobs of node n, (n^y), following some of the jobs in M as follows. List M is separated into a string of consecutive blocks upstream of n followed by a string of consecutive blocks downstream of n. Thus, the assumed topology of the forest admits a policy lg which schedules the string of consecutive blocks upstream of n prior to the block n^y. Repeated application of Cases I and II concludes the argument.

Case I. Suppose $h_n \geqq \bar{v}(M)$. We show that g can be improved by the right-modification $rg = n^{y+z} \,\|M\|\, n^{z'} \,\|\, M'''$. The comparison of g and rg requires some notation: let t_0 be the instant the jobs of n^y are completed under g. Let τ_M and τ_z be the processing times of M and n^z respectively, and let $s_M = \boldsymbol{E}\{\exp(-\alpha\tau_M)\}$, $S_n^z = \boldsymbol{E}\{\exp(-\alpha\tau_z)\}$. Assume $z' > 0$ since in the case $z' = 0$, rg saves a switch and thus is even more favorable than g. Since policies rg and g are identical during $[0, t_0)$ and $(t_0 + \tau_M + \tau_z, \infty)$, the advantage of rg is:

$$\begin{aligned} R(rg) - R(g) &= \boldsymbol{E}\{\exp(-\alpha t_0)[h_n\alpha^{-1}(1 - S_n^z) \\ &\quad + \exp(-\alpha\tau_z)\bar{v}(M)\alpha^{-1}(1 - s_M) - K_n \exp(-\alpha(\tau_z + \tau_M)) \\ &\quad - \bar{v}(M)\alpha^{-1}(1 - s_M) - \exp(-\alpha\tau_M)(h_n\alpha^{-1}(1 - S_n^z) - K_n)]\} \\ &= \boldsymbol{E}\{\exp(-\alpha t_0)\}[\alpha^{-1}(1 - S_n^z)(1 - s_M)(h_n - \bar{v}(M)) + K_n s_M(1 - S_n^z)] \\ &> 0, \end{aligned}$$

where we have used the independence of all service periods and $h_n \geqq \bar{v}(M)$. If g is such that $x_n = y + z$, then the resulting policy rg serves node n exhaustively at t_0. Otherwise, we proceed by modifying policy rg according to either Case I or Case II described below.

Case II. Suppose $h_n < \bar{v}(M)$. The argument for this case is more delicate. In accordance with (3.9), write M in block form as

$$M = B_1 \,\|B_2\|\, \cdots \,\|\, B_q \qquad \text{for some } q \in \mathbb{N},$$

where $B_i = j(i)^{u(i)}$ and adjacent blocks describe different queues. Define p to be the *least* element of $\{1, 2, \cdots, q-1\}$ such that

$$\bar{v}(B_{p+1} \,\|B_{p+2}\|\, \cdots \,\|\, B_q) \leqq h_n. \tag{3.21}$$

It is possible that p does not exist, in which case let $M^\ell = M$ and $M^r = \varnothing$; otherwise define

$$M^\ell = B_1 \,\|B_2 \cdots \|\, B_p, \tag{3.22}$$

$$M^r = B_{p+1} \,\|B_{p+2} \cdots \|\, B_q. \tag{3.23}$$

Since M^r is the largest right-tail substring of M possessing reward rate less than or

equal to h_n, Lemma 1 and the hypothesis $h_n < \bar{\nu}(M)$ reveal that the remaining left-tail substring, M^ℓ, has reward rate greater than h_n. From M^ℓ we recursively construct two strings, $M^\ell_{\mathrm{U}}(i^*)$ and $M^\ell_{\mathrm{D}}(i^*)$, so as to correspond to the jobs of M^ℓ in $\{1, 2, \cdots, n-1\}$ and $\{n+1, n+2, \cdots, N\}$ respectively, that is, jobs upstream and downstream of n, respectively. The recursion proceeds as follows.

Let $M^\ell_{\mathrm{U}}(0) = M^\ell$ as defined in (3.22) and $M^\ell_{\mathrm{D}}(0) = \varnothing$. The following properties hold for each stage i of the recursion beginning with $i = 0$:

(P1) The sequence $(1, 2, \cdots, p)$ possesses a subsequence $(m(1), m(2), \cdots, m(\hat{p}))$ such that for all $l = 1, 2, \cdots, \hat{p}$

$$\bar{\nu}(B_{m(l)} \| B_{m(l+1)} \| \cdots \| B_{m(\hat{p})}) > h_n \tag{3.24}$$

and $M^\ell_{\mathrm{U}}(i) = B_{m(1)} \| B_{m(2)} \| \cdots \| B_{m(\hat{p})}$.

(P2) $M^\ell_{\mathrm{D}}(i)$ is such that $\bar{\nu}(M^\ell_{\mathrm{D}}(i)) \leqq h_n$ (where $\bar{\nu}(\varnothing) \triangleq 0$).

(P3) For policy $g(i) \triangleq n^y \| M^\ell_{\mathrm{U}}(i) \| M^\ell_{\mathrm{D}}(i) \| M^r \| n^{z+z'} \| M'''$, $R(g(i)) \geqq R(g)$.

Property P1 expresses the condition that there is a subsequence of indices of the blocks of M^ℓ such that the resulting string $M^\ell_{\mathrm{U}}(i)$ has the property that *all* right-tail substrings possess a reward rate greater than h_n. For $i = 0$, P1 holds with $\hat{p} = p$ and $m(l) = l$ for $l = 1, 2, \cdots, p$ since (3.24) follows from the combination of Lemma 1, the Case II hypothesis, and the definition of p. Since P2 and P3 are true for $i = 0$, the basis for recursion is established.

Assume P1, P2, and P3 at stage i; we now prove them for $i+1$. By definition, $M^\ell_{\mathrm{U}}(i) = B_{m(1)} \| B_{m(2)} \| \cdots \| B_{m(\hat{p})}$. Let $k \in \{1, 2, \cdots \hat{p}\}$ be such that $j(m(k)) > n$ and $j(m(l)) < n$ for all $l \in \{k+1, k+2, \cdots, \hat{p}\}$, where $j(m(l))$ denotes the type of job served in block $B_{m(l)}$. The subsequence index k picks out the last block of $M^\ell_{\mathrm{U}}(i)$ corresponding to service in $\{n+1, n+2, \cdots, N\}$. If k does not exist, our construction is complete and $i^* = i$; otherwise, we define strings $M^\ell_{\mathrm{U}}(i, l)$ and $M^\ell_{\mathrm{U}}(i, r)$ such that $M^\ell_{\mathrm{U}}(i) = M^\ell_{\mathrm{U}}(i, l) \| B_{m(k)} \| M^\ell_{\mathrm{U}}(i, r)$. Equation (3.24) implies

$$\bar{\nu}(M^\ell_{\mathrm{U}}(i, r)) > h_n, \tag{3.25}$$

and $j(m(k)) > n$ implies by Corollary 1 that

$$\bar{\nu}(B_{m(k)}) < h_n. \tag{3.26}$$

Let

$$M^\ell_{\mathrm{U}}(i+1) \triangleq M^\ell_{\mathrm{U}}(i, l) \| M^\ell_{\mathrm{U}}(i, r)$$

$$M^\ell_{\mathrm{D}}(i+1) \triangleq B_{m(k)} \| M^\ell_{\mathrm{D}}(i).$$

For stage $i+1$, P1 follows by setting the new subsequence equal to the previous one with element k deleted. Lemma 2 implies (3.24): all right-tail strings of $M^\ell_{\mathrm{U}}(i+1)$ are improved with respect to reward rate by the deletion of block $B_{m(k)}$. Since by construction $M^\ell_{\mathrm{D}}(i+1)$ contains only jobs in queues of $\{n+1, n+2, \cdots, N\}$, P2 follows by Corollary 1. Consider

$$g(i+1) = n^y \| M^\ell_{\mathrm{U}}(i+1) \| M^\ell_{\mathrm{D}}(i+1) \| M^r \| n^{z+z'} \| M'''.$$

Since $B_{m(k)}$ does not contain descendants of $M_U^{\ell}(i, r)$, interchanging the order of $B_{m(k)}$ and $M_U^{\ell}(i, r)$ is feasible and Lemma 3 establishes P3 by (3.25) and (3.26). Policy $g(i+1)$ is non-anticipative, because we have assumed deterministic node connections.

As previously noted, the construction of $M_U^{\ell}(\cdot)$ and $M_D^{\ell}(\cdot)$ is completed at stage i^* such that $M_U^{\ell}(i^*)$ no longer contains jobs in $\{n+1, n+2, \cdots, N\}$. Define policy $lg = M_U^{\ell}(i^*) \| n^{\nu} \| M_D^{\ell}(i^*) \| M^{r} \| n^{z+z'} \| M'''$. Note that $\bar{v}(M_U^{\ell}(i^*)) > h_n$ by P1. Since $\bar{v}(n^{\nu}) < h_n$, Lemma 3 yields $R(lg) - R(g(i^*)) > 0$; thus, we conclude by P3 that $R(lg) - R(g) > 0$. We proceed further by modifying policy lg according to either Case I or Case II.

The preceding two cases show that any non-exhaustive policy g can be strictly improved by an exhaustive policy. Such an exhaustive policy can be found by first replacing g by either rg or lg and then repeatedly applying the constructive arguments of Cases I and II to points of non-exhaustion.

Thus, Theorem 1 is proven for $\alpha > 0$. When $\alpha = 0$, we establish the result using a continuity argument. Since we have assumed service periods of finite mean, standard arguments establish that the objective function $J(\cdot)$ is continuous in α at $\alpha = 0$. Continuity guarantees that an exhaustive policy which is optimal in the neighborhood of zero will remain optimal for $\alpha = 0$.

We conclude the proof of Theorem 1 with the following counterexample to the optimality of an exhaustive policy when (3.20) is not true.

Example 1. Consider the case of two queues with $c_1 = 3$, $c_2 = 4$, $K_1 = K_2 = K = 1$, $\alpha = 0$, $I(1) = 2$, $I(2) = 3$, $x_1 = 2$, $x_2 = 0$, and deterministic one unit service periods in both nodes. Thus the first node feeds the second and possesses a lesser holding cost rate than the second. Only two policies are admissible: let the first, g, serve the queues in the order 1 2 1 2 and the second, e, in the order 1 2. Since the total cost of g is $4c_1 + 2c_2 + 4K = 24$ and that of e is $3c_1 + 4c_2 + 2K = 27$, we find that exhaustive service cannot be optimal in general for problems with connections.

Remark on self-loops. The assumption of a deterministic connection from node n to $I(n)$ can be relaxed slightly without complication. Theorem 1 remains valid if a job completed at n is sent to $I(n)$ with probability p_n, $0 < p_n < 1$ (independent of all else) and reenters node n otherwise. Because it suffices to consider pure Markov policies, a policy that chooses a job in node n for a given state will continue to process that job until it eventually enters node $I(n)$. This problem can thus be reduced to a deterministic version for which the service distribution reflects the total time required for the job to reach node $I(n)$. Thus, it suffices to modify S_n and μ_n. The resulting problem with deterministic interconnections is easily calculated to possess an effective mean service rate of $\mu_n(1 - p_n)$. Problems with more general routing remain open.

Remark on the switching cost problem. For the case of no discounting ($\alpha = 0$),

the switching cost problem is trivialized under the condition given in Theorem 1, and a particular exhaustive strict priority rule is optimal. With $h_1 \geqq h_2 \geqq \cdots \geqq h_N$ and $I(n) > n$, consider a policy that exhaustively serves the queues in order $1, 2, \cdots, N$. Following Nain (1989), we call this policy the 'exhaustive upstream-queues-first $c'\mu$ rule'. The reward rate $c'\mu$ can be defined by (3.7) as $\lim_{\alpha \to 0} \alpha v_n(u) = (c_n - c_{I(n)})\mu_n$. Since each queue is visited only once, switching cost is minimized. To see that holding cost is also minimized, suppose that the jobs originally in the system and all their descendants are available for service at time $t = 0$. It is well known that holding cost is minimized by serving at each instant a job/queue maximizing $c_i'\mu_i \triangleq h_i$. Since the exhaustive upstream-queues-first $c'\mu$ rule minimizes both switching cost and holding cost and is admissible for the forest topology assumed, it is optimal. In light of the degenerate case for problems with switching cost and $\alpha = 0$, we emphasize the importance and greater difficulty of the undiscounted problem with switching delay.

The policy $(1, 2, \cdots, N)$ is not in general optimal when $\alpha > 0$ as the following counterexample shows. Consider two queues in parallel. Let $x_1 = 1$, $x_2 = 2$, $K_1 = 1$, $K_2 = 1{\cdot}5$, $c_1 = c_2$, and let the service times be equal, deterministic, one unit at both queues. Let $\alpha = 0{\cdot}1054$ so that $S_i = \exp(-\alpha\sigma_i) = 0{\cdot}9$ for $i = 1, 2$. We need consider only the exhaustive policies $g^1 = 1, 2, 2$ and $g^2 = 2, 2, 1$. Since the holding costs are identical under both policies,

$$\begin{aligned} J(g^1) - J(g^2) &= [K_1 - \exp(-\alpha\sigma_1)K_2] - [K_2 + \exp(-2\alpha\sigma_2)K_1] \\ &= [1 + 0{\cdot}9(1{\cdot}5)] - [1{\cdot}5 + 0{\cdot}81] \\ &> 0, \end{aligned}$$

even though $h_1 = h_2$ and $K_1 < K_2$.

Remark on the switching delay problem. We introduced at the end of Section 2 the case of switching delay. Although the problem with switching delay is in some respects more difficult (see, for example, the preceding remark on the undiscounted case), it shares with the problem with switching cost the structural property described in Theorem 1. The previous approach applies to the case of delays, provided the basic reward rate definitions are modified as described in Remark 2 of Section 3.1 to account for the set-up delays. To see that the case of switching delays is essentially similar to the case of switching costs, consider the following. For the discounted case with switching cost, we addressed the problem of maximizing the reward $R(g)$ given by (3.3). To compare the rewards earned under different policies, we found the use of reward rates to be effective (see (3.6)–(3.8), (3.11) and their use in the proof of Theorem 1, explicitly in Case I and via Lemma 3 in Case II). The case of switching delays is similar, but now the reward and reward rate expressions must take into account the switching delays specified by the policy under consideration. Because the reward rates are used to compare the rewards earned by an initial policy and another locally perturbed version of it, the arguments presented in the proof of Theorem 1 apply directly. Theorem 1 holds as stated under the

criterion of (2.2). The proof of Theorem 1 concludes with the following counterexample.

Example 2. Consider the problem of Example 1, except with deterministic set-up delays of $\frac{1}{2}$ unit at each node instead of the set-up costs. Because $\bar{J}(e) = 32$ while $\bar{J}(g) = 30$, exhaustive service is not optimal.

3.3. *Two nodes in tandem.* In this section, we consider the case of $N = 2$ nodes. We analyze this problem for two reasons: (1) The results for networks with two nodes can be used to (sequentially) simplify the search for optimal policies for more complicated networks (this will be shown in Section 3.4). (2) For networks with two nodes, we can extend the formulation of Section 2 to incorporate stochastic connections.

Throughout Section 3.3 we suppose that node 1 feeds node 2. Let p denote the probability that a job served at node 1 is routed to 2; with probability $1 - p$ that job exits the system. Successive routing outcomes are i.i.d. The remainder of the formulation of Section 2 applies, except that the set of pure Markov policies is now richer than G^L. Feedback policies must be considered because the state evolves subject to random routing. We show, however, that for a certain class of two-node problems, we can explicitly define an exhaustive *list* policy that is optimal. Before proceeding, note that one may also consider the case where a job served in node 2 is routed back to queue 2 with probability p_2 and exits the system with probability $1 - p_2$. Indeed, a self-loop can also be introduced at node 1. Our results hold for such problems. The self-loops are best treated as previously described in the remark on self-loops above, so in what follows we restrict attention to networks without self-loops (equivalently, forests). For the sake of applications, we conclude this section with a treatment of the undiscounted switching delay problem.

The reward and reward rate notions developed in Section 3.1 apply here as well. For $\alpha > 0$, let

$$h_1 = (c_1 - pc_2)S_1(1 - S_1)^{-1}, \tag{3.27}$$

$$h_2 = c_2 S_2 (1 - S_2)^{-1}. \tag{3.28}$$

We begin with a lemma.

Lemma 4. Suppose that node 1 feeds node 2. If $h_1 \geqq h_2$, an optimal policy must be exhaustive. If $h_1 < h_2$, an optimal policy is not, in general, exhaustive.

Proof. We first consider the case $\alpha > 0$ and argue that node 1 must be served exhaustively. The arguments of Theorem 1 can then be applied to node 2 based on the exhaustive service of node 1. The result holds for $\alpha = 0$ by the continuity argument made for Theorem 1.

We show that any policy which does not serve node 1 exhaustively can be strictly improved. Suppose that at time t, a pure Markov policy g has just completed a job in node 1 and the new queue lengths are $x(1)$ and $x(2)$ in nodes 1 and 2 respectively, where $x(1)>0$, $x(2)>0$. Moreover, assume policy g switches to node 2 at time t. Since all jobs served at 2 exit the system, it is known at time t that a non-random number of jobs, u_2, will be served in node 2 under g prior to the first return to node 1. Let $\tilde{g}$ denote the following (right) modification of g. Policy $\tilde{g}$ is identical to g on $[0, t)$, and at t serves an additional job in node 1 during $[t, t+\sigma_1)$. At time $t+\sigma_1$, $\tilde{g}$ switches to 2 and serves u_2 jobs for T ($T = f_{2,u_2}$) units of time, after which instant $\tilde{g}$ is coupled to g and the two policies thereafter possess identical costs. The reward and reward rate notions developed in Section 3.1 apply here, since h_1 incorporates the stochastic connection from node 1 to 2. If $x(1)=1$, $\tilde{g}$ saves a switch at time $t+\sigma_1+T$; thus, the advantage of policy $\tilde{g}$ over g can be expressed according to the following inequality:

$$\begin{aligned} R(\tilde{g}) - R(g) &\geqq \boldsymbol{E}\{e^{-\alpha t}\}[h_1\alpha^{-1}(1-S_1) + S_1 v_2(u_2)\alpha^{-1}(1-S_2^{u_2}) - S_1 S_2^{u_2} K_1 \\ &\quad -(v_2(u_2)\alpha^{-1}(1-S_2^{u_2}) + S_2^{u_2}(h_1\alpha^{-1}(1-S_1) - K_1))] \\ &= \boldsymbol{E}\{e^{-\alpha t}\}[K_1 S_2^{u_2}(1-S_1) + \alpha^{-1}(1-S_1)(1-S_2^{u_2})(h_1 - v_2(u_2))] \\ &> 0, \end{aligned}$$

since $v_2(u_2) < h_2 \leqq h_1$. Repetition of the above right modification $x(1)$ times yields an improved policy at each step, and eventually a policy that serves node 1 exhaustively.

Since it suffices to consider policies that serve node 1 exhaustively, the stochastic routing from node 1 is no longer an issue. The argument of Theorem 1 of Van Oyen et al. (1992) can be used to prove that it is optimal to serve node 2 exhaustively. For the sake of insight and continuity with Section 3.2, we choose instead to sketch the argument of Section 3.2, which is more general. Suppose g does not serve node 2 exhaustively; that is, g begins in 2, switches to 1 and clears it, and then returns to clear 2. Suppose $h_2 > \bar{v}(1^{x_1})$ where $\bar{v}(1^{x_1})$ is defined by (3.7). Then the arguments of Theorem 1 show that policy $rg = 2^{x_2}1^{x_1}2^{z'}$ strictly improves g, where z' indicates the number of jobs of node 1 sent to 2. On the other hand, if $h_2 \leqq \bar{v}(1^{x_1})$, then policy $lg = 1^{x_1}2^{x_2+z'}$ strictly improves g. Note that rg and lg are non-anticipative since g serves node 1 exhaustively. The following example concludes the proof by showing a case for which no exhaustive policy is optimal.

Example 3. Consider the problem specified in Example 1 except that $I(1) = 2$ with probability $\frac{1}{2}$ and jobs leave the system with probability $\frac{1}{2}$. Policy e is as before, and analogously to Example 1, g is defined to be the policy that gives strict priority to jobs in node 2. Then $J(e) - J(g) = \frac{1}{4}(2c_2 - c_1 - 2K) + \frac{1}{4}(c_2 - c_1 - K) = \frac{3}{4}$.

In the case that $h_1 \geqq h_2$, Lemma 4 limits the class of candidates for optimality to

only two policies: the first, say g^1, clears node 1 then 2; the other, g^2, clears them in the order 2 1 2. As previously noted, the case $h_1 \geqq h_2$ with $\alpha = 0$ is degenerate: policy 1 2 is always optimal. For the case $h_1 \geqq h_2$ with $\alpha > 0$, define the following indices using (3.27) and (3.28):

$$\Psi_1(x_1) \triangleq \nu_1(x_1) = h_1 - \alpha K_1/(1 - S_1^{x_1}), \tag{3.29}$$

$$\begin{aligned}\Psi_2(x_1, x_2) \triangleq\ & h_2 - \alpha K_2/(1 - S_2^{x_2}) \\ & - \alpha K_2(1 - (1-p)^{x_1})S_1^{x_1}S_2^{x_2}/((1 - S_1^{x_1})(1 - S_2^{x_2})).\end{aligned} \tag{3.30}$$

By computing the expected cost difference of g^1 and g^2, we show that $\Psi_1(x_1)$ and $\Psi_2(x_1, x_2)$ define an optimal policy. The computation is simplified by noting that along any sample path, g^1 and g^2 are identical following the second return to node 2 under g^2. Thus,

$$\begin{aligned}R(g^1) - R(g^2) &= [\nu_1(x_1)\alpha^{-1}(1 - S_1^{x_1}) + S_1^{x_1}(-K_2 + h_2\alpha^{-1}(1 - S_2^{x_2}))] \\ &\quad - [\nu_2(x_2)\alpha^{-1}(1 - S_2^{x_2}) + S_2^{x_2}\nu_1(x_1)\alpha^{-1}(1 - S_1^{x_1}) \\ &\quad - (1 - (1-p)^{x_1})S_1^{x_1}S_2^{x_2}K_2] \\ &= \alpha^{-1}(1 - S_1^{x_1})(1 - S_2^{x_2})[\Psi_1(x_1) - \Psi_2(x_1, x_2)].\end{aligned}$$

Consequently, we have derived the following result.

Theorem 2. Consider the class of problems with $N = 2$, $\alpha > 0$, $\boldsymbol{P}(I(1) = 2) = p$, $\boldsymbol{P}(I(1) = 3) = 1 - p$, $I(2) = 3$, and $h_1 \geqq h_2$. If $\Psi_1(x_1) > \Psi_2(x_1, x_2)$, it is optimal to serve the nodes exhaustively in the order 1 2; otherwise, the exhaustive policy serving the nodes in the order 2 1 2 is optimal.

Note that the index of node 2 depends on the states of both queues. Thus, even in the case of two tandem queues, the optimal policy cannot be properly regarded as an index rule of the Gittins type, because the index of one queue (project) depends on the state of another (see Gittins (1989)).

We conclude this section with a treatment of the problem of switching delay with no discounting, since we deem this case to be of significant importance in applications. We assume that independent of all else, a finite-mean random set-up delay D_n is incurred upon any switch into node n. Similar to (3.12), for the case $\alpha = 0$ we define $h_1 = (c_1 - pc_2)\mu_1$ and $h_2 = c_2\mu_2$. Because the discounted cost criterion converges to the undiscounted criterion as $\alpha \searrow 0$, the arguments of Lemma 4 indicate that an exhaustive policy is optimal, provided $h_1 \geqq h_2$. Thus, we proceed to define indices $\bar{\Psi}_1(x_1)$ and $\bar{\Psi}_2(x_1, x_2)$ similar to those in Theorem 2 so as to explicitly define an optimal rule. After some careful accounting, and the

cancellation of identical costs under the two strategies, we find that the advantage of policy 1 2 over 2 1 2 is given by

$$\begin{aligned}
&\boldsymbol{E}\{c_1x_1(\tau_2+D_2)-c_2x_2(\tau_1+D_1)-c_2px_1\tau_2\}\\
&\quad=(c_1-pc_2)x_1\boldsymbol{E}\{\tau_2+D_2\}-c_2x_2\boldsymbol{E}\{\tau_1+D_1\}-c_2px_1\boldsymbol{E}\{D_2\}\\
&\quad=\boldsymbol{E}\{(\tau_1+D_1)(\tau_2+D_2)\}[(c_1-pc_2)\mu_1\boldsymbol{E}\{\tau_1\}/E\{\tau_1+D_1\}\\
&\qquad-c_2\mu_2(\boldsymbol{E}\{\tau_2\}/\boldsymbol{E}\{\tau_2+D_2\}-c_2px_1\boldsymbol{E}\{D_2\}/E\{\tau_1+D_1\}\boldsymbol{E}\{\tau_2+D_2\})].
\end{aligned}$$

Thus, for the case of $\alpha=0$ and switching delays, the analog of Theorem 2 holds true: Provided $(c_1-pc_2)\mu_1\geqq c_2\mu_2$, policy 1 2 is optimal if $\bar{\Psi}_1(x_1)\geqq\bar{\Psi}_2(x_1,x_2)$ and 2 1 2 is optimal otherwise, where

$$\bar{\Psi}_1(x_1)=(c_1-pc_2)\mu_1^{-1}(x_1\mu_1^{-1}/(x_1\mu_1^{-1}+\boldsymbol{E}\{D_1\})), \tag{3.31}$$

$$\begin{aligned}
\bar{\Psi}_2(x_1,x_2)=c_2\mu_2[&x_2\mu_2^{-1}/(x_2\mu_2^{-1}+\boldsymbol{E}\{D_2\})\\
&-px_1\mu_2^{-1}\boldsymbol{E}\{D_2\}/((x_1\mu_1^{-1}+\boldsymbol{E}\{D_1\})(x_2\mu_2^{-1}+\boldsymbol{E}\{D_1\}))].
\end{aligned} \tag{3.32}$$

This concludes our treatment of two tandem queues. The solution of trees with more than two nodes and probabilistic interconnections, however, remains an open problem because the approach used here does not carry over to the general case.

3.4. *Computational considerations.* For the problems formulated in Section 2, the class of feasible list strategies, G^{L}, may be of extremely large cardinality depending on the number of nodes, the forest topology, and the initial queue lengths. Throughout this section, we assume the network parameters satisfy $h_1\geqq h_2\geqq\cdots\geqq h_N$ and $I(n)>n$ for $n=1,2,\cdots,N$. Thus Theorem 1 guarantees that the search may be reduced (very significantly if the queue lengths are large) to G^{E}, the set of exhaustive list policies. To be consistent with our development thus far, we emphasize the discounted problem with switching cost. We discuss the problem with switching delays only in cases where the two problems are significantly different. Although (3.7), (3.8), and (3.10) explicitly define the cost of any policy in G^{E} and thereby provide the basis for a search, the problem of determining an optimal policy remains computationally expensive. In this section we investigate two approaches that simplify the search over G^{E} for an optimal policy. First, we demonstrate how the basic reward rate expressions $\bar{\nu}(\cdot)$ and the result for two tandem queues can be used in searching G^{E}. Second, we investigate a heuristic for constructing a policy for a multitree forest network from the rules that are optimal for each constituent tree taken in isolation.

We begin with a discussion of the applicability of the basic reward rates $\bar{\nu}(\cdot)$. Consider a network of two unconnected queues, 1 and 2, which contain a and b jobs respectively. As we showed in Van Oyen et al. (1992), it is optimal to serve node 1 first if, and only if, $\bar{\nu}(1^a)\geqq\bar{\nu}(2^b)$. Such tests based on $\bar{\nu}(\cdot)$ remain valid for nodes served consecutively, provided that the order of service can be feasibly interchanged. That is, consider a forest satisfying (3.20) and a policy

$g = M_0 \| M_1 \| M_2 \| M_3 \in G^E$ such that switches are required at the beginning of both M_1 and M_2 ($M_0 = \varnothing$ and $M_3 = \varnothing$ are permitted). If $g' = M_0 \| M_2 \| M_1 \| M_3$ is a feasible policy and $\bar{v}(M_2) > \bar{v}(M_1)$, then the cost of policy g' is strictly less than that of g. Moreover, if $g' \notin G^E$, then g' can be improved by another policy $g'' \in G^E$. One may be able to construct g'' either by using again the basic reward rate expressions or by using our result for two tandem queues, and thus the comparison of g and g'' can be made without explicitly computing $J(g)$ or $J(g'')$. To illustrate the preceding discussion, consider a forest with $N > 3$ nodes, suppose nodes 1, 2, and 3 are in series, and assume queue lengths a, b and c respectively. Let $g = M_0 \| 2^b \| 3^{b+c} \| 1^a \| 2^a \| 3^a \| M_3 \in G^E$, $M_1 = 2^b \| 3^{b+c}$, and $M_2 = 1^a \| 2^a \| 3^a$. If $\bar{v}(M_2) > \bar{v}(M_1)$, then policy g cannot be optimal, and the result for two tandem queues can easily be used to determine which of the following two policies improves g the most: policy $\tilde{g} = M_0 \| 1^a \| 2^{a+b} \| 3^{a+b+c} \| M_3$ or policy $\hat{g} = M_0 \| 1^a \| 3^c \| 2^{a+b} \| 3^{a+b} \| M_3$. To see this, notice that the simple comparison $\bar{v}(M_2) > \bar{v}(M_1)$ reveals that g is more costly than $g' = M_0 \| M_2 \| M_1 \| M_3 = M_0 \| 1^a \| 2^a \| 3^a \| 2^b \| 3^{b+c} \| M_3$. Furthermore, since $g' \notin G^E$, it can be improved either by $\tilde{g}$ or by $\hat{g}$. In the following example, we elaborate on these ideas and show how we can systematically simplify the search for an optimal policy in the case of a network with three nodes.

Example 4. Consider the case of three nodes in series with $h_1 \geqq h_2 \geqq h_3$, and let node 1 feed 2 and 2 feed 3. In this case G^E contains the following nine exhaustive lists, simply denoted by the order in which queues are served:

1 2 3,	1 3 2 3,	2 1 2 3,
2 1 3 2 3,	2 3 1 2 3,	3 1 2 3,
3 2 1 2 3,	3 2 1 3 2 3,	3 2 3 1 2 3,

In this example, every feasible list is linked to two other policies by the test for two tandem queues or by the index of (3.7). Theorem 2 can be used (setting $p = 1$) to compare the following pairs of exhaustive policies:

(1 2 3, 1 3 2 3);	(1 2 3, 2 1 2 3);
(2 1 2 3, 2 1 3 2 3);	(2 3 1 2 3, 3 2 3 1 2 3);
(3 2 1 3 2 3, 3 2 1 2 3);	(3 2 1 2 3, 3 1 2 3).

For example, begin with policy 1 2 3. If $\Psi_3(x_2 + x_1, x_3) > \Psi_2(x_1 + x_2)$ (where $\Psi_3(\cdot, \cdot)$ and $\Psi_2(\cdot)$ are defined analogously to (3.30) and (3.29) respectively), then 1 2 3 is improved by 1 3 2 3.

Moreover, the basic reward rate expression of (3.7) can be used to compare other pairs of list strategies:

(1 3 2 3, 3 1 2 3); (2 1 3 2 3, 2 3 1 2 3); (3 2 3 1 2 3, 3 2 1 3 2 3).

For example, if $v_3(x_3) > v_1(x_1)$, then 1 3 2 3 is improved by 3 1 2 3.

The preceding example illustrates how the index theorem for two queues in tandem can be used to simplify the search for an optimal policy in a three-node network. It may be possible to use the results for three-node networks to simplify the search for four-node networks and so on. We believe that the computational effort depends on the topology of the forest under consideration. For example, the parallel queue topology is the simplest case and possesses a closed-form index rule, while the cardinality of G^{E} is maximized in the case of N queues in series.

In the remainder of this section, we investigate a heuristic approach that reduces the search over G^{E}. We assume that the problem of scheduling a tree can be solved, and discuss the possibility of scheduling a forest of multiple trees using lists that are optimal for the individual trees. Suppose that the forest contains l trees $T_1, T_2, \cdots, T_l$ with list M_i being optimal for tree T_i (taken in isolation). As a means of dramatically reducing the search over G^{E} (which grows quickly in l), consider restricting attention to (forest) lists which serve the trees in a manner consistent with M_i for all i. Under the assumption that policy g sequences jobs in tree i in a manner such that the ordering of M_i is preserved, the problem reduces to optimally splicing together the lists $M_1, M_2, \cdots, M_l$ so as to minimize cost (maximize reward). Because each tree is completely independent of the others, using the results of Section 3.1 we can reduce the problem to a multi-armed bandit problem treated by Gittins (1989). At any decision instant, let $B_i(1)B_i(2)\cdots B_i(q)$ denote the q remaining stages of service for tree i (according to M_i) with $B_i(j)$ denoting the jth block of jobs to be served. An index rule is optimal under the decomposition assumption, where the index of tree T_i is given by

$$\psi_i = \max_{1 \leqq j \leqq q} \bar{v}(B_i(1)B_i(2)\cdots B_i(j)). \tag{3.33}$$

Note that the switching costs (or delays) are absorbed in $\bar{v}(\cdot)$, thereby yielding a problem without switching costs. Thus, for the problem of scheduling a forest of l trees, the assumption that the optimal sequence of an isolated tree be maintained in the schedule for the forest results in substantial simplification. Upon solution of l single-tree problems, the index rule for the forest can be computed by the standard methods used for multi-armed bandit problems. If at time $t = 0$, ψ_i is maximum, the index rule proceeds to serve the first j blocks of M_i (j is defined by (3.33)). Then ψ_i is re-evaluated based on the remaining list and the procedure continues in this manner. This re-evaluation, and hence the entire policy, can be determined off-line.

Since the index rule of (3.33) is optimal under the assumption that attention be restricted to $M_1, M_2, \cdots, M_l$, we consider the removal of this assumption. The following example demonstrates that such an index rule is not optimal without the restrictive assumption made earlier.

Example 5. Consider a forest with two trees and set-up cost K for all nodes. Tree

1 contains two nodes in tandem: node 1 feeds node 2. The second tree contains an isolated node, call it queue 3. Suppose $\alpha > 0$ and $h_1 = h_2 = h_3 = h$; thus the problem reduces to that of minimizing the total expected discounted cost due only to switching, because the expected holding cost is equal under all policies in G^E. Let $x_1 = x_2 = x_3 = 1$, $S_1 = \frac{9}{10}$, $S_2 = \frac{1}{2}$, $S_3 = \frac{3}{5}$, where S_i denotes the expected discounted processing time of node i. The proposed heuristic considers the two trees in isolation. If tree 1 is isolated, a quick calculation reveals that policy 1 2 is optimal. So, the heuristic assigns the index ψ_1 (Equation (3.33)) based only on the consideration of either serving node 1 only, or node 1 followed by node 2. Using Equations (3.7) and (3.11), we get $\psi_1 = h - (2{\cdot}452)\alpha K$, which is achieved by serving node 1 followed by 2. For the second tree (queue 3), the index of Equation (3.33) is $h - (2{\cdot}500)\alpha K$. Thus, the heuristic algorithm specifies policy 1 2 3, which incurs a total switching cost of $2{\cdot}125K$. For the forest, however, policy 2 3 1 2 is optimal and incurs a total expected switching cost of $2{\cdot}070K$. The optimal list for the forest serves tree 1 in the order 2 1 2 even thought this order is not optimal for tree 1 taken separately. For this example, the heuristic gives the second best performance of seven candidates in G^E.

Example 5 can be explained as follows. Because $h_1 = h_2 = h_3$, and $S_2 < S_3 < S_1$, it follows that the reward rate of a single job in node 2 is greater than that for node 3, which is in turn greater than that for node 1. When scheduling tree 1, the large reward rate of serving node 2 does not offset the penalty of an additional switch, thus 1 2 is optimal. For the forest, however, the time required to serve job 3 sufficiently discounts the cost of the second switch into 2 so that it is optimal to follow 2 3 1 2 and thereby maximize reward rate.

Example 5 relied heavily on a large discount factor (or possibly a small α with very long service periods) to identify a case for which the optimal forest-schedule diverged from a policy known to be optimal for one of the constituent trees. Since either policy 1 2 3 or 3 1 2 is optimal for $\alpha = 0$ under the condition given in Theorem 1, the forest scheduling problem with switching costs can be decomposed without loss of optimality in this case. For the problem with switching delay, however, the problem is more complex. Even in the case of $\alpha = 0$, it is not optimal to construct the schedule for the forest based on optimal lists for the trees, as the following example indicates.

Example 6. Consider a forest consisting of three nodes. Node 1 deterministically feeds node 2, forming a tree of tandem queues. Node 3 is unconnected. For all $i = 1, 2, 3$ let $D_i = \mu_i = 1$. Let $\alpha = 0$, $c_1 = \frac{5}{2}$, $c_2 = 1$, $c_3 = \frac{2}{5}$, $x_1 = 1$, $x_2 = 9$, and leave x_3 unspecified. Thus, we have $h_1 = (c_1 - pc_2)\mu_1 = \frac{3}{2}$, $h_2 = c_2\mu_2 = 1$, $h_3 = c_3\mu_3 = \frac{2}{5}$, and Theorem 1 guarantees that an exhaustive policy is optimal. Theorem 2 as applied to the problem of switching delay gives $\bar{\Psi}_2(x_1, x_2) = \frac{17}{20} > \bar{\Psi}_1(x_1) = \frac{3}{4}$ and so policy 2 1 2 is an optimal policy for nodes 1 and 2 considered in isolation. The heuristic algorithm determines from (3.33) and (3.19) that the index of the first tree, ψ_1,

equals $\frac{9}{10}$ and is achieved by serving 9 jobs in node 2. Because the index of queue 3 is $0{\cdot}4x_3/(x_3+1) < \frac{2}{5}$, the algorithm begins service in node 2. Moreover, the algorithm specifies policy 2 1 2 3 for any value of x_3. With respect to the holding cost incurred in node 3 alone, however, policy 1 2 3 saves cost c_3x_3 over 2 1 2 3, because it requires one less switch. Since the holding cost incurred in nodes 1 and 2 is greater under policy 1 2 than policy 2 1 2 by amount 16 cost units, we find that policy 2 1 2 3 is optimal for $x_3 \leqq 40$ and 1 2 3 is optimal for $x_3 > 40$.

This example shows that if one restricts attention to policy 2 1 2 (which is optimal for that tree) when scheduling the forest, the result is suboptimal for $x_3 > 40$ by amount $c_3x_3 - 16$. This loss is unbounded in x_3. It is, however, bounded as a fraction of the total cost incurred under an optimal policy. For this example the fractional loss incurred by considering only schedules based on the 2 1 2 strategy is at most $1/(2 + (x_3+1)/(2\mu_3 \boldsymbol{E}\{D\})) < \frac{1}{2}$.

4. Conclusion

We have analyzed the optimal stochastic scheduling of a single server in systems of queues connected as forests and for which switching of the server is penalized. Although exhaustive service is not optimal in general, we identified in Theorem 1 a class of forests for which exhaustive service is optimal. This limited characterization of optimal policies contrasts sharply with the elegant index policies that are optimal when switching penalties are not modelled. In Theorem 2 we explicitly define an optimal policy for a class of networks of two tandem queues with stochastic routing. We indicated that this policy is different from the Gittins index policy. Finally, we presented some strategies for simplifying the search for an optimal policy in systems of the class identified in Theorem 1.

Acknowledgements

The authors thank the referee for many suggestions that have improved the content and clarity of this paper.

References

AGRAWAL, R., HEGDE, M., AND TENEKETZIS, D. (1988) Asymptotically efficient adaptive allocation rules for the multi-armed bandit problem with switching cost. *IEEE Trans. Autom. Control* **33,** 899–906.

AGRAWAL, R., HEGDE, M., AND TENEKETZIS, D. (1990) Multi-armed bandit problems with multiple plays and switching cost. *Stoch. Stoch. Rep.* **29,** 437–459.

BROWNE, S. AND YECHIALI, U. (1989) Dynamic priority rules for cyclic-type queues. *Adv. Appl. Prob.* **21,** 432–450.

BRUNO, J. AND DOWNEY, P. (1978) Complexity of task sequencing with deadlines, set-up times and changeover costs. *SIAM J. Comput.* **7,** 393–404.

DEMPSTER, M. A. H., LENSTRA, J. K., AND RINNOOY KAN, A. M. G. (1982) *Deterministic and Stochastic Scheduling.* Reidel, Dordrecht.

FOSS, S. G. (1984) Queues with customers of several types. In *Advances in Probability Theory*: *Limit Theorems, and Related Problems,* ed. A. A. Borovkov, pp. 348–377. Springer-Verlag, New York.

GITTINS, J. C. (1979) Bandit processes and dynamic allocation indices. *J. R. Statist. Soc. B* **41,** 147–177.

GITTINS, J. C. (1989) *Multi-armed Bandit Allocation Indices.* Wiley, New York.

GITTINS, J. C. AND JONES, D. M. (1974) A dynamic allocation index for the sequential design of experiments. Read at the 1972 European Meeting of Statisticians, Budapest. In *Progress in Statistics* ed. J. Gani et al., pp. 241–266. North-Holland, Amsterdam.

GLAZEBROOK, K. D. (1980) On stochastic scheduling with precedence relations and switching cost. *J. Appl. Prob.* **17,** 1016–1024.

GLAZEBROOK, K. D. AND GITTINS, J. C. (1981) On single-machine scheduling with precedence relations and linear or discounted costs. *Operat. Res.* **29,** 161–173.

GUPTA, D., GERCHAK, Y., AND BUZACOTT, J. A. (1987) On optimal priority rules for queues with switchover costs, Preprint, Dept. of Management Sciences, University of Waterloo.

HARRISON, J. M. (1975) Dynamic scheduling of a multi-class queue: discount optimality. *Operat. Res.* **23,** 270–282.

HOFRI, M. AND ROSS, K. W. (1987) On the optimal control of two queues with server setup times and its analysis. *SIAM J. Comput.* **16,** 399–420.

JO, K. Y. (1987) Decomposition approximation of queueing-network control models with tree structures. *Ann. Operat. Res.* **8,** 117–132.

KLIMOV, G. P. (1974) Time sharing service systems I. *Theory Prob. Appl.* **19,** 532–551.

KLIMOV, G. P. (1978) Time sharing service systems II. *Theory Prob. Appl.* **23,** 314–321.

LAI, T. L. AND YING, Z. (1988) Open bandit processes and optimal scheduling of queueing networks. *Adv. Appl. Prob.* **20,** 447–472.

LIU, Z., NAIN, P., AND TOWSLEY, D. (1992) On optimal polling policies. *QUESTA* **11,** 59–84.

MAGNANTI, T. L. AND VACHANI, R. (1990) A strong cutting plane algorithm for production scheduling with changeover costs. *Operat. Res.* **38,** 456–473.

MONMA, C. L. AND POTTS, C. N. (1989) On the complexity of scheduling with batch set up times. *Operat. Res.* **37,** 798–804.

NAIN, P. (1989) Interchange arguments for classical scheduling problems in queues. *Systems Control Lett.* **12,** 177–184.

NAIN, P., TSOUCAS, P., AND WALRAND, J. (1989) Interchange arguments in stochastic scheduling. *J. Appl. Prob.* **27,** 815–826.

PERKINS, J. R. AND KUMAR, P. R. (1989) Stable distributed real-time scheduling of flexible manufacturing/assembly/disassembly systems. *IEEE Trans. Autom. Control* **34,** 139–148.

RAJAN, R. AND AGRAWAL, R. (1991) Stochastic dominance in homogeneous queueing systems with switchover costs. Preprint.

ROSS, S. (1983) *Introduction to Stochastic Dynamic Programming.* Academic Press, New York.

SANTOS, C. AND MAGAZINE, M. (1985) Batching in single operation manufacturing systems. *Operat. Res. Lett.* **4,** 99–103.

VAN OYEN, M. P. (1992) Optimal Stochastic Scheduling of Queueing Networks: Switching Costs and Partial Information. Ph.D. Thesis, University of Michigan.

VAN OYEN, M. P., PANDELIS, D., AND TENEKETZIS, D. (1992) Optimality of index policies for stochastic scheduling with switching penalties. *J. Appl. Prob.* **29,** 957–966.

VARAIYA, P., WALRAND, J., AND BUYUKKOC, C. (1985) Extensions of the multi-armed bandit problem. *IEEE Trans. Autom. Control* **30,** 426–439.

WALRAND, J. (1988) *An Introduction to Queueing Networks.* Prentice-Hall, Englewood Cliffs, NJ.

WEBER, R. R. AND WEISS, G. (1990) On an index policy for restless bandits. *J. Appl. Prob.* **27,** 637–648.

WHITTLE, P. (1988) Restless bandits: activity allocation in a changing world. In *A Celebration of Applied Probability,* ed. J. Gani, *J. Appl. Prob.* **25A,** 287–298.