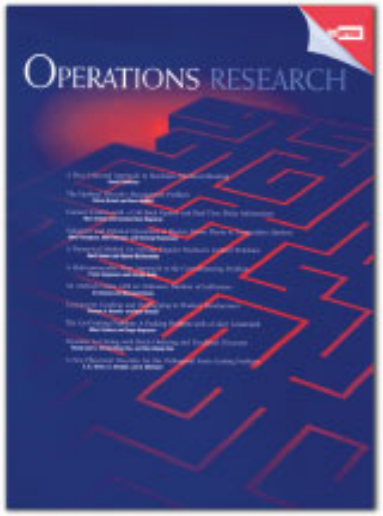

Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Stochastic Scheduling on Parallel Machines Subject to Random Breakdowns to Minimize Expected Costs for Earliness and Tardy Jobs

Xiaoqiang Cai, Sean Zhou,

To cite this article:
Xiaoqiang Cai, Sean Zhou, (1999) Stochastic Scheduling on Parallel Machines Subject to Random Breakdowns to Minimize Expected Costs for Earliness and Tardy Jobs. Operations Research 47(3):422-437. http://dx.doi.org/10.1287/opre.47.3.422

STOCHASTIC SCHEDULING ON PARALLEL MACHINES SUBJECT TO RANDOM BREAKDOWNS TO MINIMIZE EXPECTED COSTS FOR EARLINESS AND TARDY JOBS

XIAOQIANG CAI

The Chinese University of Hong Kong, Shatin, Hong Kong

SEAN ZHOU

The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong

(Received May 1995; revisions received November 1996, April 1997; accepted April 1997)

This paper addresses a stochastic scheduling problem in which a set of independent jobs are to be processed by a number of identical parallel machines under a common deadline. Each job has a processing time, which is a random variable with an arbitrary distribution. Each machine is subject to stochastic breakdowns, which are characterized by a Poisson process. The deadline is an exponentially distributed random variable. The objective is to minimize the expected costs for earliness and tardiness, where the cost for an early job is a general function of its earliness while the cost for a tardy job is a fixed charge. Optimal policies are derived for cases where there is only a single machine or are multiple machines, the decision-maker can take a static policy or a dynamic policy, and job preemptions are allowed or forbidden. In contrast to their deterministic counterparts, which have been known to be NP-hard and are thus intractable from a computational point of view, we find that optimal solutions for many cases of the stochastic problem can be constructed analytically.

In this article we examine a stochastic scheduling model, which is to schedule n jobs with random processing times on m parallel identical machines subject to stochastic breakdowns, so as to minimize the expected costs for earliness and tardiness of job completions about a common deadline. The processing times may have any arbitrary distributions. The breakdowns of each machine follow a Poisson process. The deadline, denoted as D, is an exponentially distributed random variable. The tardiness cost for job i is a constant w_i, which is incurred once job i is completed after D. The earliness cost for job i is a general function $g_i(X)$, where $X = D - C_i$ and C_i is the completion time of job i. For brevity, we use the notation $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ to denote this model (cf., e.g., Blazewicz, et al. 1994), where Pm stands for m parallel identical machines, U_i is an indicator variable that takes value 1 if job i is tardy and 0 otherwise, and $\mathrm{E}(Y)$ represents the expectation of a random variable Y.

Applications of the model are varied. An example is in agriculture. In countries such as Australia where long periods of drought occur from time to time, planning properly based on rain forecast is important. Suppose that a farmer is to plant crops on a large number of stretches of land that require sufficient rainwater in the early stage of growth. The timing of the next rainfall, however, is quite uncertain and can only be estimated based on weather forecast information. Planting too early before it rains could lower the crop yields and even jeopardize their growth. Planting after the rain, however, could lead to big losses, as it is quite likely that no more rain will come before the end of the current planting season. A limited number of planters are available, which could break down stochastically. The amount of time required to complete planting on a stretch is not known in advance, and the arrival time of the next rainfall is, of course, a random variable. How to plan the planting so as to minimize the expected total loss is a problem such as we are modelling.

Another application is in manufacturing. Suppose that a number of jobs with random processing times are to be processed in a manufacturing system and then delivered by a transporter to customers. The arrival (or available) time of the transporter, however, is not under the control of the manufacturer, and is a random variable to the manufacturer. On one hand, completion of a job after the arrival of the transporter will incur a much higher delivery cost because an alternative, substantially more expensive means will have to be used to deliver a job if it misses the transporter. The parameter w_i reflects the extra cost incurred to transport such a job i. On the other hand, a job, if completed before the arrival of the transporter, will have to be stored in a warehouse to wait for the transporter. This incurs an inventory cost $g_i(X)$, which is a function of the waiting time X. Clearly, the problem to minimize the expected total cost in this manufacturing scenario also falls into our model. In general, the model is applicable wherever a number of tasks should be completed before an external event that occurs with uncertainty, but early completions are not desirable, and the stochastic nature of resource availability requires a careful sequencing of the tasks in accordance with their relative importance, the

Subject classifications: Production/scheduling, sequencing: earliness/tardiness, multiple machines. Production/scheduling, stochastic: random processing times, machine breakdowns, deadline.
Area of review: MANUFACTURING OPERATIONS.

0030-364X/99/4703-0422 $05.00

possible processing times, and the probability of the occurrence of the event.

The model we propose is a stochastic scheduling problem with a nonregular performance measure (cf. Baker and Scudder 1990), which differs from traditional problems in its inclusion of earliness costs. When earliness penalties are ignored, problems involving minimization of the number of tardy jobs have been studied extensively in the literature. Blau (1973) examined a single-machine model where due dates are random and distinct. Balut (1973), Kise et al. (1982), and Kise and Ibaraki (1983) considered the problem of minimizing the number of tardy jobs with probability not smaller than a given level, when processing times are normally distributed and due dates are constants. Katoh and Ibaraki (1983) studied a similar problem, except that all jobs have a common due date. Derman et al. (1978) considered processing times with exponential distributions and a common due date with an arbitrary distribution. Pinedo (1983) derived several results when jobs are exponentially distributed, due dates follow a same distribution, and optimal policies are static or dynamic. Boxma and Forst (1986) investigated problems where there is a single-machine or a flow shop consisting of multiple machines. Emmons and Pinedo (1990) examined multimachine problems with due dates under various assumptions. De et al. (1991) considered general random processing times and a common, exponentially distributed due date. Sarin et al. (1991) minimized the expected sum of tardy probabilities under a common, deterministic due date. Recently Coffman et al. (1993) considered an exponentially distributed deadline, which they call a *waiting timer*. When the timer expires, any job not yet processed will require an additional machine to process and the objective is to minimize the expected number of machines needed. Note that the interpretation of a random deadline as a random waiting timer also applies to our model.

Scheduling problems with machines subject to stochastic breakdowns, when earliness costs are not considered, have also received limited attention. Pinedo and Ross (1980) investigated a single-machine problem in which the machine is subject to external shocks according to a nonhomogeneous Poisson process. Glazebrook (1984) examined a single-machine problem with machine breakdowns and formulated it as a cost-discounted Markov decision process. Birge et al. (1990) considered more general breakdown processes. Allahverdi and Mittenthal (1994) showed that a problem with parallel machines subject to random breakdowns can be converted to an equivalent deterministic parallel-machine problem with modified job processing times, when the problem is to minimize the expected mean flow time and the breakdowns follow a generalized Poisson process.

The work of this article is in line with the recent trend of research on earliness/tardiness (E/T) scheduling. Our concentration, however, is on the effects of stochastic phenomena in such a problem. Research on probabilistic phenomena is still scanty in this relatively new scheduling field. Some relevant works are reviewed below. Chakravarthy (1986), Vani and Raghavachari (1987), and Cai (1996) investigated the completion time variance problem with random processing times, which is equivalent to an E/T problem of minimizing the sum of squared deviations (SSD) of job completion times from a common due date. Cheng (1986, 1987) studied a problem with random processing times where the due dates are restricted by some prespecified rules. Soroush and Fredendall (1994) and Cai and Zhou (1996) recently addressed the problem of minimizing the sum of absolute deviations of completion times when the processing times are normally distributed. Mittenthal and Raghavachari (1993) considered a single-machine problem with a SSD measure, in which the processing times are deterministic but the machine is subject to stochastic breakdowns. They established properties of optimal static policies when machine breakdowns are governed by a class of Poisson processes.

There is a commonality observed in the works reviewed above, namely, the stochastic E/T problems are generally more complex and difficult than their deterministic counterparts. The results found in this paper on $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$, however, contrast sharply with this observation. We note that a deterministic version of $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ has been examined by Kahlbacher and Cheng (1993). They have shown that many cases of this deterministic problem are NP-hard, either in the ordinary sense or in the strong sense. These results indicate the intractability of the problem when problem attributes are deterministic. In contrast to these results, we shall show, in this article, that many cases of $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ are well solvable. Optimal policies for various cases will be derived analytically. In particular, in the single-machine case, we shall show that an optimal static policy should partition the jobs into two sets, with jobs in the first set being started at time zero and sequenced according to a simple rule, and jobs in the second set being started after the deadline in arbitrary order. Optimal dynamic policies will also be obtained when preemptions are allowed or forbidden. In the multimachine case, we shall show that the optimal static policies assign and sequence the jobs on the machines in a batch by batch manner, under certain simple rules. Dynamic policies will also be examined.

The remainder of the paper is organized as follows: In Section 1 the model is described together with the assumptions. Section 2 addresses the single-machine problem, where a deterministic equivalent of the objective function is derived first, then followed by constructions of optimal static and dynamic policies. Section 3 examines the multimachine problem, where optimal static and dynamic policies are derived for various cases. Finally, some concluding remarks are given in Section 4.

1. MODEL AND ASSUMPTIONS

We consider a problem in which a set, $\mathcal{N} = \{1, 2, \ldots, n\}$, of independent jobs must be processed on a set, $\mathcal{M} = \{1, 2, \ldots, m\}$, of parallel identical machines under a

common deadline D, which is an exponential random variable with parameter δ. Each job can be processed by any one of the machines. The amount of time needed to process job i, $i \in \mathcal{N}$, denoted as P_i, is a random variable with an arbitrary distribution, which is independent of other jobs and of the manner in which the job is assigned to a machine. At time zero, all jobs are ready for processing and all machines are assumed to be operable. However, a machine will be subject to stochastic breakdowns after it starts processing jobs. The breakdown process of machine j is characterized by a sequence of finite-valued, positive random vectors $\{Y_{jk}, Z_{jk}\}_{k=1}^{\infty}$, where Y_{jk} is the duration of the kth uptime of machine j and Z_{jk} is the duration of the kth downtime of machine j.

Assume that the breakdown process of each machine is independent of other machines, that the uptimes of a machine are independent of its downtimes, and that the downtimes are mutually independent and identically distributed. Let z_j be the starting time of machine j (the time when it starts its first job). For the stochastic sequence $\{Y_{jk}\}_{k=1}^{\infty}$, define a counting process $\{N_j(t) : t \geq z_j\}$ as follows:

$$N_j(t) = \sup\{k \geq 0 : S_{jk} \leq t\}, \tag{1.1}$$

where S_{jk} denotes the total uptime of machine j before its kth breakdown, namely,

$$S_{jk} = \sum_{i=1}^{k} Y_{ji}, \quad k \geq 1. \tag{1.2}$$

For convenience we also define $S_{j0} = 0$.

We consider the situation where $N_j(t)$, $\forall j \in \mathcal{M}$, is a Poisson process with rate τ. In addition, we are concerned with a *preempt-resume* model (cf. Birge et al. 1990), i.e., if a machine breakdown occurs during the processing of a job, the work done on the job prior to the breakdown is not lost, and the processing of the disrupted job can be continued from where it was interrupted. Thus, the processing times P_i, $i \in \mathcal{N}$, are independent of any machine breakdowns.

Both static and dynamic policies will be examined. Under a static policy, decisions on the processing of all the jobs should be determined a priori and maintained throughout until all jobs are finished. We assume that, when a static policy is concerned, job preemptions are forbidden, namely, if a job has been started, its processing must be continued until completion. Under a dynamic policy, the decision maker may revise his decisions dynamically without being bound to any decisions made before. Job preemptions may be allowed or forbidden. At any moment in time, the decision maker will specify a job for processing on each operable machine, by taking into account all information that has become available up to the moment.

It is clear that the completion time of a job i, for $i \in \mathcal{N}$, depends upon the policy adopted. Write a policy as ζ, and the corresponding completion time of job i under ζ as $C_i(\zeta)$. Our problem is to determine an optimal policy, within the class of (static or dynamic) policies that can be adopted, to minimize the expectation of the total cost for earliness and tardy jobs, namely,

$$\mathrm{ETC}(\zeta) = \mathrm{E}\Big[\sum_{C_i(\zeta) \leq D} g_i(D - C_i(\zeta)) + \sum_{C_i(\zeta) > D} w_i\Big], \tag{1.3}$$

where $g_i(\cdot)$ and w_i are, respectively, the earliness function and the fixed tardiness penalty of job i as defined before.

2. SINGLE-MACHINE CASE

2.1. Objective Function Development

In this subsection, we shall derive a deterministic equivalent of the objective function (1.3) for the problem where the number of machines is one and the decision policies that can be taken are static. This result will serve as the basis of all subsequent analyses. Note that in the development below we shall drop, without ambiguity, the reference index j associated with the counting process $N_j(t)$ and the downtime duration Z_{jk} as now there is only one machine.

In general, it has been known that, to minimize an earliness/tardiness performance measure, it might be desirable for a machine to be idle for a period of time before processing a job (cf. Baker and Scudder 1990). Therefore, inserted idle times are decision variables that should be optimally determined in earliness/tardiness problems. In our problem, let s_i, $i \in \mathcal{N}$, denote the amount of time that the machine is kept idle immediately before job i is started. (Note that s_i does not include the downtimes of the machine, namely, only uptimes are counted in our definition.) For the time being, we assume that s_i is a deterministic variable, taking value in $[0, +\infty]$. Later we will see that a policy with s_i approaching $+\infty$ is equivalent, in terms of minimizing the overall cost, to another policy with a random idle time depending on D (see Item 1 of Remark 2).

In the single-machine case, a static policy should specify the sequence to process the jobs on the machine, and the idle time s_i inserted immediately before each job i. Let $\lambda = (\lambda(1), \lambda(2), \ldots, \lambda(n))$ be a job sequence, which denotes that job i is the kth job to be processed if $i = \lambda(k)$, and let $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ be a set of n elements. Hence, a static policy can be expressed as $\zeta = \{\lambda, \mathcal{S}\}$. One can see that the completion time of job i, under ζ, can be written as

$$C_i(\zeta) = R_i(\zeta) + \sum_{k=0}^{N(R_i(\zeta)-)} Z_k, \tag{2.1}$$

where $R_i(\zeta)$ represents the total uptime of the machine before job i is finished and

$$Z_0 \equiv 0. \tag{2.2}$$

Moreover, because we have assumed that no preemptions are allowed in the static case, $R_i(\zeta)$ can be further expressed as

$$R_i(\zeta) = \sum_{k \in \mathcal{B}_i(\lambda)} (s_k + P_k), \tag{2.3}$$

where $\mathcal{B}_i(\lambda)$ represents the set of jobs sequenced no later than job i under ζ. Note that the starting time z_j of the machine (see Section 1) is equal to the inserted idle time $s_{\lambda(1)}$ associated with the first job to be processed on the machine.

Theorem 1 gives an equivalent form of the earliness/tardiness criterion (1.3). As we shall see later, the result will be utilized in deriving various optimal policies, including those for the multi-machine case to be examined in Section 3. The criterion given in Theorem 1 involves three new parameters, α_i, f_k, and η. These parameters did not appear in the original form of the earliness/tardiness problem, but they are very important to the solutions for the problem. We will provide, in Remark 1 (to be given after the proof of Theorem 1), some interpretations on these parameters, from which we can better see their relevance to the characteristics of our earliness/tardiness problem. The equivalent criterion (2.4) is also explained in Remark 1.

Theorem 1. *When ζ is a static policy, then*

$$\mathrm{ETC}(\zeta) = \sum_{i=1}^{n} (\alpha_i - w_i) \prod_{k\in\mathcal{B}_i(\lambda)} e^{-\eta s_k} f_k + \sum_{i=1}^{n} w_i, \tag{2.4}$$

where

$$\alpha_i = \int_0^{+\infty} g_i(y)\delta e^{-\delta y}\, dy, \tag{2.5}$$

$$f_k = \mathrm{E}[e^{-\eta P_k}], \tag{2.6}$$

$$\eta = \delta + \tau \Pr(D \leqslant Z), \tag{2.7}$$

and Z is a random variable whose distribution is the same as the common distribution of the downtimes Z_k.

Proof. Rewrite the objective function (1.3) as

$$\begin{aligned}\mathrm{ETC}(\zeta) &= \mathrm{E}\Big[\sum_{i=1}^{n} g_i(D - C_i(\zeta))I_{\{C_i(\zeta)\leqslant D\}} \\ &\qquad + \sum_{i=1}^{n} w_i I_{\{C_i(\zeta)>D\}}\Big] \\ &= \sum_{i=1}^{n} \{\mathrm{E}[g_i(D - C_i(\zeta))I_{\{C_i(\zeta)\leqslant D\}}] \\ &\qquad + w_i \Pr(C_i(\zeta) > D)\},\end{aligned} \tag{2.8}$$

where I_A denotes the indicator of an event A, which equals 1 if A occurs and 0 otherwise.

Let us now examine $\Pr(D > C_i(\zeta))$. Noting that as D is exponentially distributed, we have $\Pr(D > X_1 + X_2) = \Pr(D > X_1)\Pr(D > X_2)$ for any nonnegative random variables X_1, X_2 such that D, X_1, X_2 are mutually independent. Thus, it follows from the mutual independence between D, $N(t)$, Z_k and P_k that

$$\begin{aligned}&\Pr(D > C_i(\zeta)|R_i(\zeta) = x) = \Pr\Big(D > x + \sum_{k=0}^{N(x-)} Z_k\Big) \\ &= \Pr(D > x)\Pr(D > Z_1 + Z_2 + \cdots + Z_{N(x-)}) \\ &= e^{-\delta x} \sum_{k=0}^{\infty} \Pr(D > Z_1 + \cdots + Z_k)\Pr(N(x-) = k).\end{aligned} \tag{2.9}$$

Since the counting process of machine breakdowns, $N(t)$, is a Poisson process with rate τ, (2.9) becomes

$$\begin{aligned}&\Pr(D > C_i(\zeta)|R_i(\zeta) = x) \\ &= e^{-\delta x} \sum_{k=0}^{\infty} [\Pr(D > Z)]^k \frac{(\tau x)^k}{k!} e^{-\tau x} \\ &= e^{-\delta x} e^{\tau x \Pr(D>Z)} e^{-\tau x} = e^{-(\delta - \tau\Pr(D>Z) + \tau)x} \\ &= e^{-(\delta + \tau\Pr(D\leqslant Z))x} = e^{-\eta x}.\end{aligned} \tag{2.10}$$

Noting that $\Pr(A) = \mathrm{E}[\Pr(A|X)]$ for any event A and random variable X according to double expectation formula, (2.10) leads to

$$\Pr(D > C_i(\zeta)) = \mathrm{E}\{\Pr(D > C_i(\zeta)|R_i(\zeta))\} = \mathrm{E}[e^{-\eta R_i(\zeta)}]. \tag{2.11}$$

By (2.3) and the independence between P_k, $k \in \mathcal{N}$, we have

$$\mathrm{E}[e^{-\eta R_i(\zeta)}] = \prod_{k\in\mathcal{B}_i(\lambda)} e^{-\eta s_k} \mathrm{E}[e^{-\eta P_k}] = \prod_{k\in\mathcal{B}_i(\lambda)} e^{-\eta s_k} f_k. \tag{2.12}$$

Hence (2.11) reduces to

$$\Pr(D > C_i(\zeta)) = \prod_{k\in\mathcal{B}_i(\lambda)} e^{-\eta s_k} f_k. \tag{2.13}$$

Now conditional on $C_i(\zeta) = x$, we can calculate

$$\begin{aligned}&\mathrm{E}[g_i(D - C_i(\zeta))I_{\{C_i(\zeta)\leqslant D\}}|C_i(\zeta) = x] \\ &= \mathrm{E}[g_i(D - x)I_{\{x\leqslant D\}}] \\ &= \int_x^{\infty} g_i(t - x)\delta e^{-\delta t}\, dt = \int_0^{\infty} g_i(y)\delta e^{-\delta(x+y)}\, dy \\ &= e^{-\delta x}\int_0^{\infty} g_i(y)\delta e^{-\delta y}\, dy = \alpha_i e^{-\delta x}.\end{aligned} \tag{2.14}$$

Furthermore,

$$\begin{aligned}&\Pr(D > C_i(\zeta)|C_i(\zeta) = x) \\ &= \Pr(D > x|C_i(\zeta) = x) = \Pr(D > x) = e^{-\delta x}.\end{aligned} \tag{2.15}$$

It follows from (2.11), (2.14) and (2.15) that

$$\begin{aligned}&\mathrm{E}[g_i(D - C_i(\zeta))I_{\{C_i(\zeta)\leqslant D\}}] \\ &= \alpha_i \mathrm{E}[e^{-\delta C_i(\zeta)}] = \alpha_i \mathrm{E}[\Pr(D > C_i(\zeta)|C_i(\zeta))] \\ &= \alpha_i \Pr(D > C_i(\zeta)) = \alpha_i \mathrm{E}[e^{-\eta R_i(\zeta)}].\end{aligned} \tag{2.16}$$

In addition, by (2.11),

$$\Pr(C_i(\zeta) > D) = 1 - \Pr(C_i(\zeta) \leqslant D) = 1 - \mathrm{E}[e^{-\eta R_i(\zeta)}]. \tag{2.17}$$

Substituting (2.16) and (2.17) into (2.8), and using (2.12), we have (2.4). □

From the proof above, we can make the following observations:

Remark 1.

(1) One can see from (2.11) that $\Pr(C_i(\zeta) < D) = \mathrm{E}[e^{-\eta R_i(\zeta)}] = \Pr(R_i(\zeta) < D')$, where $D' = (\delta/\eta)D$. This implies that the machine breakdown process actually has the effect of reducing the deadline D to an earlier deadline

D', since $\eta \geqslant \delta$ by (2.7). This is understandable because during the periods of breakdowns the machine is not usable. The "shortened" deadline D' is also exponentially distributed, and the parameter η is its rate.

(2) According to (2.13) and item (1) above, the multiplication in (2.4) can be expressed as:

$$\prod_{k\in\mathcal{B}_i(\lambda)} e^{-\eta s_k} f_k = \Pr(R_i(\zeta) < D') = \Pr(C_i(\zeta) < D),$$

which can be viewed as the probability that job i is completed before the "shortened" deadline D' when $R_i(\zeta)$ is regarded as the completion time, or before the real deadline D when the real completion time $C_i(\zeta)$ is considered.

(3) It is easy to see from (2.6) that $f_i = \mathrm{E}[e^{-\eta P_i}] = \Pr(P_i < D')$. Thus, the parameter f_i represents the probability that the processing time of job i is less than the deadline D' (or the probability that job i is not tardy if it is started at time zero). This is a parameter related to the characteristics of job i. As we will see in the subsequent sections, these parameters play an essential role in determining the optimal job sequence λ^*.

(4) Item (2) above together with (2.4) shows that

$$\mathrm{ETC}(\zeta) = \sum_{i=1}^{n} \{\alpha_i \Pr(C_i(\zeta) < D) + w_i \Pr(C_i(\zeta) > D)\}. \tag{2.18}$$

This is another equivalent form of the objective function (1.3). By the memoryless property of D, or comparing (2.18) with (2.8), it can be seen that $\alpha_i = \mathrm{E}[g_i(D - C_i(\zeta))|C_i(\zeta) < D]$ under any ζ, which may be viewed as the earliness cost of job i under the condition that it is early. This explains (2.18), and consequently its equivalent (2.4).

2.2. Static Policies

Let us use $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ to denote the problem when there is only a single machine. The deterministic counterpart of the problem is optimized by a static sequence. However, finding this optimal sequence is an NP-hard problem even in the special case with $g_i(\cdot) \equiv 0$ for all $i \in \mathcal{N}$ (which is equivalent to the Knapsack problem; see Karp 1972). We now show that an optimal static policy for the stochastic problem $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ can be obtained analytically. This result is summarized in the following theorem, in which $\mathcal{I}$ and $\mathcal{I}^c$ are defined such that $\alpha_i < w_i$ if $i \in \mathcal{I}$ and $\alpha_i \geqslant w_i$ if $i \in \mathcal{I}^c$.

Theorem 2. *An optimal static policy for the stochastic problem* $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *will process the jobs according to the following rules*:

(a) *Jobs in* $\mathcal{I}$ *are sequenced in nonincreasing order of* $(w_i - \alpha_i)f_i/(1 - f_i)$ *and processed from time zero with no idle time inserted between any two consecutive jobs*;

(b) *All jobs in* $\mathcal{I}^c$ *are processed in an arbitrary order, starting as soon as* D *has occurred or the last job in* $\mathcal{I}$ *has been completed, whichever is the later.*

Proof. According to (2.4), one can see that $\mathrm{ETC}(\zeta)$ is minimized when s_i is zero if $w_i > \alpha_i$ and infinity if $w_i \leqslant \alpha_i$. This implies that, under an optimal policy, jobs in $\mathcal{I}$ should be processed from time zero with no inserted idle time between any two consecutive jobs, whereas jobs in $\mathcal{I}^c$ should be started as late as possible, which should not be earlier than the completions of the jobs in $\mathcal{I}$. This also implies that the sequences for the two sets of jobs can be considered separately.

We now show that Rule (a) is optimal. This can be done in a similar way as Theorem 2.1 of Boxma and Forst (1986). For completeness, we provide the arguments below:

Assume, for notational brevity, that $\mathcal{I} = \{1, 2, \ldots, n_I\}$ and consider two sequences $\lambda_I = (1, 2, \ldots, n_I)$ and $\lambda_I' = (1, \ldots, i-1, i+1, i, i+2, \ldots, n_I)$, where λ_I' is the sequence resulted from interchanging the ith and $(i+1)$th jobs in the sequence λ_I. Let ζ_I and ζ_I' be two policies which differ in the sequences λ_I and λ_I'. Then by (2.4) and noting that $s_i = 0$ for $i \in \mathcal{I}$, we have

$$\begin{aligned}
&\mathrm{ETC}(\zeta) - \mathrm{ETC}(\zeta') \\
&\quad = (\alpha_i - w_i) f_1 f_2 \ldots f_{i-1} f_i \\
&\qquad + (\alpha_{i+1} - w_{i+1}) f_1 f_2 \ldots f_{i-1} f_i f_{i+1} \\
&\qquad - (\alpha_{i+1} - w_{i+1}) f_1 f_2 \ldots f_{i-1} f_{i+1} \\
&\qquad - (\alpha_i - w_i) f_1 f_2 \ldots f_{i-1} f_{i+1} f_i \\
&\quad = (\alpha_i - w_i) f_1 \ldots f_{i-1} f_i (1 - f_{i+1}) \\
&\qquad - (\alpha_{i+1} - w_{i+1}) f_1 f_2 \ldots f_{i-1} f_{i+1} (1 - f_i) \\
&\quad = f_1 \ldots f_{i-1} (1 - f_i)(1 - f_{i+1}) \left\{ \frac{(\alpha_i - w_i) f_i}{1 - f_i} \right. \\
&\qquad\qquad \left. - \frac{(\alpha_{i+1} - w_{i+1}) f_{i+1}}{1 - f_{i+1}} \right\}.
\end{aligned}$$

Thus $\mathrm{ETC}(\zeta) \leqslant \mathrm{ETC}(\zeta')$ if and only if

$$\frac{(\alpha_i - w_i) f_i}{1 - f_i} \leqslant \frac{(\alpha_{i+1} - w_{i+1}) f_{i+1}}{1 - f_{i+1}},$$

which together with $s_i = 0$ for all $i \in \mathcal{I}$ proves Rule (a).

Now consider Rule (b). From (2.4) we can see that the contribution to the objective function from job j is given by

$$\mathrm{ETC}_j = (\alpha_j - w_j) \prod_{k\in\mathcal{B}_j(\lambda)} e^{-\eta s_k} f_k + w_j.$$

If job j is in $\mathcal{I}^c$, then $\alpha_j \geqslant w_j$ and thus $\mathrm{ECT}_j \geqslant w_j$ under any rule. This means that any policy, if it results in $\mathrm{ECT}_j = w_j$ for all $j \in \mathcal{I}^c$, must be optimal. We can see that both the following policies can achieve $\mathrm{ECT}_j = w_j$: (i) a policy with $s_j = +\infty$ for all $j \in \mathcal{I}^c$; and (ii) a policy that starts all jobs $j \in \mathcal{I}^c$ after D has occurred (because this will obviously lead to $\Pr(C_i(\zeta) > D) = 1$ and so $\mathrm{ETC}_j = w_j$). Rule (b) is proved by argument (ii) together with a feasibility consideration that jobs in $\mathcal{I}^c$ can only start as early as the last jobs in $\mathcal{I}$ is completed (whereas argument (i) leads to item (2) of Remark 2 below). □

Remark 2.

(1) Under the policy of Theorem 2, there may be an idle time between the last completed job in $\mathcal{J}$ and the first started job in $\mathcal{J}^c$, which is a random variable as it depends upon D.

(2) Note that in our model formulated here, we assume that all jobs must be processed (this may be a requirement of the contract between, say, the manufacturer and its customers). If the scheduling problem does not require that all jobs should be completed, then Rule (b) in Theorem 2 may be replaced by: *Jobs in $\mathcal{J}^c$ are not processed at all*. This remark applies to all policies to be developed in the rest of this article.

2.3. Dynamic Policies

We now address the situations where a policy can be revised dynamically. The problem is to determine, at any time t, a job for processing by taking into account all information available up to t. Clearly, in the single-machine case we need only to examine time t when that machine is up, since no new decision can be applied when the machine is down. Any t discussed in the following will mean such a time when the machine is operable.

Consider first the case where preemptions are not allowed, namely, a new decision can be applied only when a job is completed. We have the following theorem.

Theorem 3. *When preemptions are forbidden, the optimal static policy as specified by Theorem* 2 *is optimal in the class of dynamic policies for* $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$.

Proof. We use induction to prove the theorem. Our induction assertion is as follows: an optimal nonpreemptive dynamic policy should schedule at any time t the jobs which remain to be processed, according to Theorem 2. The induction will be made on the number of jobs remaining to be processed at t.

The assertion is clearly true when there is only one job remaining to be processed, since any nonpreemptive policy is a static policy if there is only one job. Now, assume that the assertion is true for any k remaining jobs. We want to prove the assertion for $k + 1$ jobs under this assumption.

If $k + 1$ jobs remain to be processed at time t, then under any policy, one job must be processed first until completion, and then followed by the other k jobs. By the induction assumption, after the first job is completed, the optimal static policy as specified by Theorem 2 is optimal dynamically for the remaining k jobs. Therefore an optimal dynamic policy on $k + 1$ jobs becomes a static policy with one job to be processed first without interruption followed by a static policy on the remaining k jobs. Thus, there are totally $k + 1$ static policies, including the one specified by Theorem 2, to be considered. The problem of determining an optimal dynamic policy reduces to a problem of determining the best policy among these $k + 1$ static policies. Clearly, if the deadline D has already occurred at or before t, then any policy, including the one specified by Theorem 2, is optimal. If the deadline D has not occurred, then because of the memoryless property of the exponential deadline D, the problem of determining the optimal static policy reduces to the situation addressed in Theorem 2, except that the number of jobs now is $k + 1$. In this case the static policy of Theorem 2 is optimal. This completes the proof. □

We now turn to the case where preemptions are allowed. In this case, the processing on a job may be interrupted, if necessary, before its completion so that another job is started.

First, recall the objective function development in Section 2.1. When ζ is a preemptive dynamic policy, (2.16) and (2.17) are still valid, but (2.12) is not (since $R_i(\zeta)$ can no longer be expressed as (2.3)). Thus, substituting (2.16) and (2.17) into (2.8), we have

$$\mathrm{ETC}(\zeta) = \sum_{i=1}^{n} (\alpha_i - w_i)\mathrm{E}[e^{-\eta R_i(\zeta)}] + \sum_{i=1}^{n} w_i, \tag{2.19}$$

where α_i and η are given by (2.5) and (2.7), respectively.

By (2.19) one can see that, to minimize $\mathrm{ETC}(\zeta)$, $R_i(\zeta)$ for $i \in \mathcal{J}$ (i.e., $\alpha_i < w_i$) should be made as small as possible, whereas $R_i(\zeta)$ for $i \in \mathcal{J}^c$ should be made as large as possible. Consequently, with respect to the jobs in $\mathcal{J}$, an optimal policy ζ^* should minimize

$$\mathrm{ETC}_{\mathcal{J}}(\zeta) = \mathrm{E}\Big(\sum_{i\in\mathcal{J}} (\alpha_i - w_i)e^{-\eta R_i(\zeta)}\Big), \tag{2.20}$$

in the class of dynamic policies, where no idle times should be inserted before any jobs in $\mathcal{J}$. Interestingly, the above happens to fall into the framework of multi-armed bandit problems, which are concerned with the question of how to allocate dynamically a single resource (machine) among a number of preemptable jobs (cf. Gittins 1989 and Weber 1992). It is known that these problems are solvable by using the Gittins Index (Gittins 1989, p. 25), which can be expressed as follows for our problem of minimizing $\mathrm{ETC}_{\mathcal{J}}(\zeta)$:

$$G_i(T_i(t)) = \sup_{\tau > T_i(t)} \frac{(w_i - \alpha_i)\displaystyle\int_{(T_i(t),\tau]} e^{-\eta s}\, dQ_i(s)}{\displaystyle\int_{(T_i(t),\tau]} (1 - Q_i(s))e^{-\eta s}\, ds}, \quad i \in \mathcal{J}, \tag{2.21}$$

where $Q_i(s)$ is the cumulative distribution function of processing time P_i and $T_i(t)$ denotes the realization of the processing time on job i up to the moment t.

In the discrete case where P_i takes integer values, the Gittins Index reduces to:

$$G_i(T_i(t)) = \max_{\tau > T_i(t)} \frac{(w_i - \alpha_i)\sum_{s=T_i(t)+1}^{\tau} e^{-\eta s}\mathrm{Pr}(P_i = s)}{\sum_{s=T_i(t)+1}^{\tau} e^{-\eta s}\mathrm{Pr}(P_i \geqslant s)}, \quad i \in \mathcal{J}. \tag{2.22}$$

To minimize $\mathrm{ETC}_{\mathcal{J}}(\zeta)$ of (2.20), a dynamic policy ζ^* should, at any time t, process the job i^* that has the maximum Gittins Index among all the unfinished jobs at time t

(see Weber 1992 for a proof of the optimality of this policy). This, together with a similar argument of Theorem 2 on processing the jobs in $\mathcal{J}^c$ (or the argument of item (2) of Remark 2), gives us Theorem 4.

Theorem 4. *When preemption is allowed, an optimal dynamic policy for the stochastic problem* $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *will process the jobs according to the following rules*:

(a) *At any time t before the occurrence of D, choose the job i* with*

$$G_{i^*}(T_{i^*}(t)) = \max_{i\in\mathcal{J}_u(t)} G_i(T_i(t)), \tag{2.23}$$

to be the present job to process, where $\mathcal{J}_u(t)$ *denotes the set of jobs which belong to* $\mathcal{J}$ *and have not been completed at time t*;

(b) *After the occurrence of D, process all unfinished jobs in an arbitrary order with no preemptions*.

Note that $G_i(T_i(t))$ depends only on information concerning job i. This greatly reduces the dimensionality of the problem of finding an optimal dynamic solution. In general, given the processing time distributions, the earliness function $g_i(x)$ and the parameter δ of the deadline D (to compute α_i according to (2.5)), and the machine breakdown distribution (to compute η according to (2.7)), one can determine the Gittins Index according to (2.21) or (2.22), and then construct an optimal dynamic policy according to Theorem 4.

A result can be further obtained under a condition on the *hazard rate* of P_i, defined as $\rho_i(s) = q_i(s)/(1 - Q_i(s))$, where $q_i(s)$ represents the density function of P_i if P_i is continuous, or the probability mass function if P_i is discrete. It is not hard to show that, if $\rho_i(s)$ is a nondecreasing function in s within the support of $q_i(s)$, then $G_i(x)$ is a nondecreasing function in x. In this case, by Theorem 4 it is clear that job i will never be preempted if it has been selected to start. Consequently, the problem reduces to one with no preemptions as addressed by Theorem 3. This immediately leads to Corollary 1.

Corollary 1. *The static policy of Theorem* 2 *remains optimal among the class of dynamic policies even though preemptions are allowed, if the hazard rates* $\rho_i(s)$ *of the processing times* P_i, *for all* $i \in \mathcal{J}$, *are nondecreasing functions in s*.

Furthermore, noting that the condition of Corollary 1 is satisfied when P_i reduce to deterministic constants p_i, we have the following result, which concerns with an important case where all processing times are deterministic but the machine is subject to stochastic breakdowns and the deadline is random.

Corollary 2. *When processing times are deterministic, the nonpreemptive policy of Theorem* 2, *where* $f_i = e^{-\eta p_i}$ *for all i, is optimal for* $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *in the classes of dynamic policies, no matter whether preemptions are allowed or not*.

It follows from Corollary 1 that, when all jobs in $\mathcal{J}$ have nondecreasing hazard rates, preemptions are not required, and the optimal policy can be determined and fixed in advance. In general, the optimal dynamic policy of Theorem 4 could, however, require preemptions. To determine which job should be processed at a time t, one should compute $G_i(T_i(t))$ at this time for all $i \in \mathcal{J}_u(t)$. This sometimes can be cumbersome, particularly for processing time distributions that are continuous and (2.21) has to be used. Next we discuss another class of dynamic policies. We will see that the optimal solution in this class of policies is easier to compute.

This class of policies, referred to as *open-loop dynamic policies* (ODP), correspond to the so-called *open-loop feedback controls* (OFC) in the dynamic control literature. An optimal control (or policy) in the class of OFC (or ODP) should minimize the expected cost starting from the current time t by taking into account the new measurements (realizations of random variables such as processing times) received up to t, under the assumption that a control adopted now would not be changed later. In other words, an optimal control in OFC should be the one that is optimal among all static controls computed at t. In contrast to OFC, an optimal control in the class of *close-loop feedback controls* (CFC) minimizes the expected cost starting from t, no matter what control will be adopted later. For detailed discussions, see Bayard (1987), Bertsekas (1976), Mayne and Michalska (1990), and Tse and Athans (1972). The dynamic policies we discussed above correspond to CFC. In what follows we will write a policy as ζ_o^* if it is optimal among the class of ODP. Note that ζ_o^* could require that a job be dynamically preempted by another job, as the job to be processed at time t depends on the realizations of processing times that are functions of t.

Because of the memoryless property of the exponential deadline D, it turns out that determining ζ_o^* is quite simple. Consider a time t. Recall the definitions of $\mathcal{J}_u(t)$ (see Theorem 4) and $T_i(t)$ (see (2.21)). Associated with each job $i \in \mathcal{J}_u(t)$, there is a revised f_i that corresponds to the remaining processing time of this job. Let the revised f_i at time t be denoted as $f_i^u(t)$. Then, by (2.6),

$$f_i^u(t) = \mathrm{E}[e^{-\eta(P_i - T_i(t))}|P_i \geqslant T_i(t)], \quad i \in \mathcal{J}_u(t). \tag{2.24}$$

If the deadline has occurred, then clearly any policy for processing the remaining jobs is optimal. Otherwise, the memoryless property of the deadline reduces the problem of determining an optimal policy in the class of ODP to one as addressed in Theorem 2 with respect to the jobs in $\mathcal{J}_u(t)$ and processing times $P_i - T_i(t)$. This gives us the following result.

Theorem 5. *When preemptions are allowed, a policy* ζ_o^* *for* $1\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *will process the jobs according to the following rules*:

(a) *At any time t before the occurrence of D, the job being processed should have the largest value of* $(w_i - \alpha_i)f_i^u(t)/(1 - f_i^u(t))$ *among all jobs in the set* $\mathcal{J}_u(t)$;

(b) *After the occurrence of D, all unfinished jobs are processed in an arbitrary order and preemption is not necessary*.

The policy ζ_o^* given by Theorem 5 is easy to compute and implement, which is valid for processing times with arbitrary distributions. We now further show that if $f_i^u(t)$ for all $i \in \mathcal{I}_u(t)$ are nondecreasing functions of t, then ζ_o^* reduces to the optimal static policy of Theorem 2 (or more precisely, the static policy of Theorem 2 becomes one of the policies implied by Theorem 5 (note that Rule (b) in Theorem 5 implies the existence of multiple solutions). This is the case if the processing times satisfy the condition (2.25) as given in the lemma below.

Lemma 1. *If X is a nonnegative random variable satisfying*

$$\Pr(X > x + y) \leqslant \Pr(X > x)\Pr(X > y), \tag{2.25}$$

for all nonnegative values x and y, then

$$\mathrm{E}[e^{-\eta(X-a)} | X \geqslant a] \geqslant \mathrm{E}[e^{-\eta X}] \quad \text{for all } a \geqslant 0. \tag{2.26}$$

Proof. Let $F(x) = \Pr(X \leqslant x)$ be the cumulative distribution function of X and, for brevity, write $h(X) = -e^{-\eta X}$. Clearly $h(\cdot)$ is a nondecreasing function such that $\mathrm{E}[|h(X)|] < \infty$. Using Fubini's Theorem we obtain

$$\begin{aligned}
\mathrm{E}[h(X)] &= \int_{x\geqslant 0} h(x)\, dF(x) \\
&= \int_{x\geqslant 0} \left[h(0) + \int_{0\leqslant y\leqslant x} dh(y) \right] dF(x) \\
&= h(0) + \int_{y\geqslant 0} \int_{y\leqslant x} dF(x)\, dh(y) \\
&= h(0) + \int_{y\geqslant 0} [1 - F(y)]\, dh(y),
\end{aligned} \tag{2.27}$$

and similarly,

$$\begin{aligned}
&\mathrm{E}[h(X-a) I_{\{X\geqslant a\}}] \\
&\quad = \int_{x\geqslant a} h(x-a)\, dF(x) = \int_{z\geqslant 0} h(z)\, dF(z+a) \\
&\quad = \int_{z\geqslant 0} \left[h(0) + \int_{0\leqslant y\leqslant z} dh(y) \right] dF(z+a) \\
&\quad = h(0)[1 - F(a)] + \int_{y\geqslant 0} [1 - F(y+a)]\, dh(y) \\
&\quad = h(0)\Pr(X > a) + \int_{y\geqslant 0} \Pr(X > y + a)\, dh(y).
\end{aligned} \tag{2.28}$$

Consequently by (2.25) and (2.27)–(2.28),

$$\begin{aligned}
&\mathrm{E}[h(X-a) | X \geqslant a] \\
&\quad = \frac{1}{\Pr(X > a)} \mathrm{E}[h(X-a) I_{\{X\geqslant a\}}] \\
&\quad = \frac{1}{\Pr(X > a)} \left\{ h(0)\Pr(X > a) \right. \\
&\qquad \left. + \int_{y\geqslant 0} \Pr(X > y + a)\, dh(y) \right\} \\
&\quad \leqslant h(0) + \frac{1}{\Pr(X > a)} \left\{ \int_{y\geqslant 0} \Pr(X > y)\Pr(X > a)\, dh(y) \right\} \\
&\quad = h(0) + \int_{y\geqslant 0} \Pr(X > y)\, dh(y) = \mathrm{E}[h(X)].
\end{aligned}$$

This is equivalent to (2.26), with $h(X) = -e^{-\eta X}$. □

By Lemma 1 we get $f_i^u(t) \geqslant f_i$ for all $t \geqslant 0$ as $T_i(t) \geqslant 0$, if P_i satisfies the condition (2.25). Therefore, if $(w_i - \alpha_i) f_i / (1 - f_i) \geqslant (w_j - \alpha_j) f_j / (1 - f_j)$, then $(w_i - \alpha_i) f_i^u(t) / (1 - f_i^u(t)) \geqslant (w_j - \alpha_j) f_j / (1 - f_j)$ for all $t \geqslant 0$. Thus, Theorem 5 gives us the following result.

Corollary 3. *The static policy of Theorem 2 remains optimal among the class of ODP even though preemptions are allowed, if P_i, for all $i \in \mathcal{I}$, satisfy the condition* (2.25).

Remark 3.

(1) The condition (2.25) is weaker than the condition of nondecreasing hazard rate required by Corollary 1. Consider an example where $\rho(s) = s$ for $0 \leqslant s \leqslant 2$ and $\rho(s) = 1$ for $s > 2$. Then obviously $\rho(s)$ is not a nondecreasing function. However, the random variable X with $\rho(s)$ as its hazard rate satisfies the condition in (2.25), which can be easily seen from the relation

$$\Pr(X > x) = e^{-\int_0^x \rho(s)\, ds}.$$

(2) Nevertheless, most well-known probability distributions which satisfy (2.25) also have nondecreasing hazard rates. Important examples (which satisfy both conditions) include *exponential distributions, uniform distributions, Erlang distributions, Gamma distributions with shape parameters* $\geqslant 1$, *Weibull distributions with shape parameters* $\geqslant 1$, *(truncated) normal distributions, Gompertz distributions*, etc.

3. MULTI-MACHINE CASE

In this section we will first investigate the multi-machine problem with $w_i \equiv w$ and $g_i(\cdot) \equiv g(\cdot)$, $\forall i \in \mathcal{N}$. The results thus obtained will then be extended to more general cases with certain compatibility conditions in Section 3.3. Note that in the multiple-machine case, the deterministic version of $Pm \| \mathrm{E}\{\Sigma(g_i + w_i U_i)\}$ has already been NP-hard in the strong sense when m is a variable, even if $w_i \equiv w$ and $g_i(\cdot) \equiv g(\cdot)$ (Kahlbacher and Cheng 1993 have shown that *multiprocessor scheduling* is reducible to this problem, whereas *multiprocessor scheduling* is NP-hard in the strong sense when m is a variable, see Garey and Johnson 1979).

We use the notation $Pm \| \mathrm{E}\{\Sigma(g + wU_i)\}$ to represent the problem with $w_i = w$ and $g_i = g$.

3.1. Static Policies

In the single-machine case, a static policy consists of a sequence λ and a set $\mathcal{S} = \{s_i\}_{i=1}^n$. In the multi-machine case considered here, what we are concerned with is a static policy ζ which specifies a priori the assignment of jobs to the machines and the schedule of the jobs on each machine. Such a static policy ζ can be described as $\zeta = \{(\lambda_j, \mathcal{S}_j)\}_{j=1}^m$, where λ_j denotes the sequence of the jobs on

machine j and $\mathcal{S}_j$ denotes the set of idle times inserted before these jobs. More explicitly, $\lambda_j = (\lambda_j(1), \ldots, \lambda_j(n_j))$ is an ordered n_j-tuple, where n_j denotes the number of jobs assigned to machine j while $\lambda_j(k) \in \mathcal{N}$ denotes that job $\lambda_j(k)$ is the kth to be processed on machine j. Accordingly, $\mathcal{S}_j = \{s_i^j : i = \lambda_j(1), \ldots, \lambda_j(n_j)\}$ with s_i^j being the idle time inserted immediately before job i on machine j. As in Section 2.1, here we assume that s_i^j are deterministic variables. Also, let $\mathcal{N}_j$ denote the set of jobs assigned to machine j.

Similar to Theorem 1 in Section 2.1, we can show that,

$$\mathrm{ETC}(\zeta) = \sum_{j=1}^{m} \sum_{i \in \mathcal{N}_j} (\alpha_i - w_i) \prod_{k \in \mathcal{B}_i(\lambda_j)} e^{-\eta s_k^j} f_k + \sum_{i=1}^{n} w_i. \tag{3.1}$$

For $Pm \| \mathrm{E}\{\Sigma(g + wU_i)\}$, the above reduces to:

$$\mathrm{ETC}(\zeta) = (\alpha - w) \sum_{j=1}^{m} \sum_{i \in \mathcal{N}_j} \prod_{k \in \mathcal{B}_i(\lambda_j)} e^{-\eta s_k^j} f_k + nw, \tag{3.2}$$

where $\alpha = \int_0^{+\infty} g(y) \delta e^{-\delta y}\, dy$ according to (2.5). Thus, if $\alpha \geqslant w$, then $\mathrm{ETC}(\zeta) \geqslant nw$ for any policy ζ. In this case, it is easy to show (cf. the proof for Rule (b) of Theorem 2) that any policy is optimal if it starts all jobs after the occurrence of D. The more interesting but difficult situation is the case where $\alpha < w$. Note that $\alpha < w$ means that the cost of missing the deadline is greater than the earliness cost (see item (4) of Remark 1), which is usually the case in practical situations. In the rest of this subsection, we will consider this case.

When $\alpha < w$, an optimal policy must have $s_k^j = 0$ for all k and j in order to minimize $\mathrm{ETC}(\zeta)$. Hence, it is clear that the problem now becomes to determine an optimal policy so as to maximize:

$$\mathrm{ETC}'(\zeta) = \sum_{j=1}^{m} \sum_{i \in \mathcal{N}_j} \prod_{k \in \mathcal{B}_i(\lambda_j)} f_k. \tag{3.3}$$

Let us now establish the following lemma, which is essential in order to get the optimal policy for maximizing $\mathrm{ETC}'(\zeta)$. The lemma establishes how one may assign and sequence a set of $2u$ elements into two ordered u-tuples $(a_1, a_2, \ldots, a_u)$ and $(b_1, b_2, \ldots, b_u)$ so that $S_u = \Sigma_{i=1}^{u} a_1 \ldots a_i + \Sigma_{i=1}^{u} b_1 \ldots b_i$ is maximized.

Lemma 2. *Let $\{a_1, a_2, \ldots, a_u\}$ and $\{b_1, b_2, \ldots, b_u\}$ be two sets of numbers in $[0, 1)$. Define $A_i = a_1 a_2 \ldots a_i$, $B_i = b_1 b_2 \ldots b_i$ for $i = 1, \ldots, u$ and $S_u = A_1 + \cdots + A_u + B_1 + \cdots + B_u$.*

If one of the following three conditions holds:

(i) *there exists a $k \in \{1, 2, \ldots, u-1\}$ such that $a_{k+1} > b_k$,*

(ii) *there exists a $k \in \{1, 2, \ldots, u-1\}$ such that $b_{k+1} > a_k$, or*

(iii) *there exist two numbers $k, l \in \{1, 2, \ldots, u\}$ such that $a_k < b_k$ and $a_l > b_l$,*

then we can rearrange $a_1, \ldots, a_u, b_1, \ldots, b_u$ to obtain two new sets $\{a'_1, \ldots, a'_u\}$ and $\{b'_1, \ldots, b'_u\}$ so that

$$S'_u = A'_1 + \cdots + A'_u + B'_1 + \cdots + B'_u > S_u, \tag{3.4}$$

where

$$A'_i = a'_1 \ldots a'_i, \quad B'_i = b'_1 \ldots b'_i. \tag{3.5}$$

Proof. First we assume that Condition (i) holds. Let $\mathcal{T} = \mathcal{T}_u$ be the subset of $\{1, 2, \ldots, u-1\}$ such that $a_{k+1} > b_k$ for all $k \in \mathcal{T}$ and $a_{k+1} \leqslant b_k$ for $k \in \mathcal{T}^c = \{1, 2, \ldots, u-1\} - \mathcal{T}$. By Condition (i) $\mathcal{T}$ is nonempty. We now regroup the numbers $a_1, \ldots, a_u, b_1, \ldots, b_u$ into two new sets $\{a'_1, \ldots, a'_u\}$ and $\{b'_1, \ldots, b'_u\}$ by defining

$$a'_{k+1} = \begin{cases} b_k & \text{if } k \in \mathcal{T} \\ a_{k+1} & \text{if } k \in \mathcal{T}^c \end{cases},$$

for $k = 0, \ldots, u-1$ and

$$b'_k = \begin{cases} a_{k+1} & \text{if } k \in \mathcal{T} \\ b_k & \text{if } k \in \mathcal{T}^c \end{cases},$$

for $k = 1, \ldots, u$ (i.e., interchange a_{k+1} with b_k for every $k \in \mathcal{T}$). Then

$$A'_i = a'_1 \ldots a'_i = a_1 \ldots a_i \prod_{k \in \mathcal{T}, k<i} \frac{b_k}{a_{k+1}},$$
$$B'_i = b'_1 \ldots b'_i = b_1 \ldots b_i \prod_{k \in \mathcal{T}, k \leqslant i} \frac{a_{k+1}}{b_k} \tag{3.6}$$

(if $b_k = 0$ for some $k \in \mathcal{T}$, then we define $b_k/b_k = 1$), where a product over an empty index set is defined as equal to 1. As $a_{k+1} > b_k$ for all $k \in \mathcal{T}$, (3.4) will be proved by the following claim:

With a'_k and b'_k as defined above, we have for all $u \geqslant 2$ that

$$S'_u - S_u > \Big(\prod_{k \in \mathcal{T}} a_{k+1} - \prod_{k \in \mathcal{T}} b_k \Big) b_1 \ldots b_u \prod_{k \in \mathcal{T}} \frac{1}{b_k}. \tag{3.7}$$

We now prove (3.7) by induction. For $u = 2$, we must have $\mathcal{T} = \{1\}$ and $a_2 > b_1$. Hence

$$\begin{aligned} S'_u - S_u &= A'_1 + A'_2 + B'_1 + B'_2 - A_1 - A_2 - B_1 - B_2 \\ &= a_1 + a_1 b_1 + a_2 + a_2 b_2 - a_1 \\ &\quad - a_1 a_2 - b_1 - b_1 b_2 \\ &= a_1(b_1 - a_2) + a_2 - b_1 + b_2(a_2 - b_1) \\ &= (a_2 - b_1)(1 - a_1 + b_2) > (a_2 - b_1) b_2. \end{aligned}$$

Thus (3.7) holds for $u = 2$. Next, suppose that (3.7) holds for a $u \geqslant 2$, and consider $u+1$ in place of u (note that $\mathcal{T}$ will denote $\mathcal{T}_{u+1}$ and $\mathcal{T}^c = \{1, \ldots, u\} - \mathcal{T}$ in the following arguments). There are two cases.

Case I. $a_{u+1} \leqslant b_u$ (i.e., $u \notin \mathcal{T}$). In this case,

$$S_{u+1} = S_u + A_{u+1} + B_{u+1} \quad \text{and}$$
$$S'_{u+1} = S'_u + A'_{u+1} + B'_{u+1}.$$

Hence by the induction assumption and (3.6),

$$
\begin{aligned}
S'_{u+1} - S_{u+1} &= S'_u - S_u + A'_{u+1} + B'_{u+1} - A_{u+1} - B_{u+1} \\
&> \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{1}{b_k} \\
&\quad + a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{b_k}{a_{k+1}} + b_1 \ldots b_{u+1} \\
&\quad \cdot \prod_{k\in\mathcal{T}} \frac{a_{k+1}}{b_k} - a_1 \ldots a_{u+1} - b_1 \ldots b_{u+1} \\
&= \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{1}{b_k} \\
&\quad + a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}} \\
&\quad \cdot \Big(\prod_{k\in\mathcal{T}} b_k - \prod_{k\in\mathcal{T}} a_{k+1}\Big) + b_1 \ldots b_{u+1} \\
&\quad \cdot \prod_{k\in\mathcal{T}} \frac{1}{b_k} \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&= \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&\quad \cdot \Big(b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k} + b_1 \ldots b_u \\
&\qquad \cdot \prod_{k\in\mathcal{T}} \frac{1}{b_k} - a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}\Big) \\
&\geqslant \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k},
\end{aligned}
$$

where the last inequality holds because by the definition of $\mathcal{T}$, $a_{k+1} > b_k$ for $k \in \mathcal{T}$ and $b_k \geqslant a_{k+1}$ for $k \in \mathcal{T}^c$ (note that $\mathcal{T}^c = \{1, \ldots, u\} - \mathcal{T}$ when $u+1$ is in place of u), so that $\Pi_{k\in\mathcal{T}} a_{k+1} > \Pi_{k\in\mathcal{T}} b_k$ and

$$
\begin{aligned}
b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{1}{b_k} &= \prod_{k\in\mathcal{T}^c} b_k \geqslant \prod_{k\in\mathcal{T}^c} a_{k+1} \\
&= a_2 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}} \\
&\geqslant a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}.
\end{aligned}
$$

Thus (3.7) holds for $u+1$ in Case I.

Case II. $a_{u+1} > b_u$ (so that $u \in \mathcal{T}$). Note that in this case, a_{u+1} is not included in S'_u. Hence

$$
S'_u = (A'_1 + B'_1) + \cdots + (A'_u + b'_1 \ldots b'_{u-1} b_u),
$$

while

$$
\begin{aligned}
S'_{u+1} &= (A'_1 + B'_1) + \cdots + (A'_u + b'_1 \ldots b'_{u-1} b'_u) \\
&\quad + (A'_{u+1} + B'_{u+1})
\end{aligned}
$$

(i.e., the interchange between a_{u+1} and b_u affects S'_{u+1}, but not S'_u). Consequently, by (3.6),

$$
\begin{aligned}
S'_{u+1} - S'_u &= b'_1 \ldots b'_{u-1} b'_u - b'_1 \ldots b'_{u-1} b_u + A'_{u+1} \\
&\quad + B'_{u+1} = b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{a_{k+1}}{b_k} - b_1 \ldots b_u \\
&\quad \cdot \prod_{k\in\mathcal{T}-\{u\}} \frac{a_{k+1}}{b_k} + A'_{u+1} + B'_{u+1}. \qquad (3.8)
\end{aligned}
$$

Moreover, the induction assumption now implies

$$
\begin{aligned}
S'_u - S_u &> \Big(\prod_{k\in\mathcal{T}-\{u\}} a_{k+1} - \prod_{k\in\mathcal{T}-\{u\}} b_k\Big) b_1 \ldots b_u \\
&\quad \cdot \prod_{k\in\mathcal{T}-\{u\}} \frac{1}{b_k}.
\end{aligned}
$$

Thus (3.8) leads to

$$
\begin{aligned}
S'_{u+1} - S_{u+1} &= S'_u - S_u + A'_{u+1} + B'_{u+1} - A_{u+1} - B_{u+1} \\
&\quad + b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{a_{k+1}}{b_k} - b_1 \ldots b_u \prod_{k\in\mathcal{T}-\{u\}} \frac{a_{k+1}}{b_k} \\
&> \Big(\prod_{k\in\mathcal{T}-\{u\}} a_{k+1} - \prod_{k\in\mathcal{T}-\{u\}} b_k\Big) b_1 \ldots b_u \prod_{k\in\mathcal{T}-\{u\}} \frac{1}{b_k} \\
&\quad + \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&\quad \cdot \Big(b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k} - a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}\Big) \\
&\quad - b_1 \ldots b_u \prod_{k\in\mathcal{T}-\{u\}} \frac{a_{k+1}}{b_k} + b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{a_{k+1}}{b_k} \\
&= -b_1 \ldots b_u + b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{a_{k+1}}{b_k} \\
&\quad + \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&\quad \cdot \Big(b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k} - a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}\Big) \\
&= b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{1}{b_k} \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&\quad + \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) \\
&\quad \cdot \Big(b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k} - a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}\Big) \\
&\geqslant \Big(\prod_{k\in\mathcal{T}} a_{k+1} - \prod_{k\in\mathcal{T}} b_k\Big) b_1 \ldots b_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{b_k},
\end{aligned}
$$

where the last inequality holds because $b_k \geqslant a_{k+1}$ for $k \in \mathcal{T}^c$ so that

$$
\begin{aligned}
b_1 \ldots b_u \prod_{k\in\mathcal{T}} \frac{1}{b_k} &= \prod_{k\in\mathcal{T}^c} b_k \geqslant \prod_{k\in\mathcal{T}^c} a_{k+1} \\
&= a_2 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}} \\
&\geqslant a_1 \ldots a_{u+1} \prod_{k\in\mathcal{T}} \frac{1}{a_{k+1}}.
\end{aligned}
$$

This shows that (3.7) holds for $u+1$ in Case II as well and completes the proof under Condition (i). The proof under Condition (ii) is similar as the roles of $\{a_i\}$ and $\{b_i\}$ are interchangeable.

It remains to prove the conclusion of the lemma under Condition (iii). By Condition (iii) there exists a nonempty subset $\mathcal{U}$ of $\{1, \ldots, u\}$ such that $a_k < b_k$ if and only if $k \in \mathcal{U}$ and $a_l > b_l$ for at least one $l \in \mathcal{U}^c = \{1, \ldots, u\} - \mathcal{U}$. Define $a'_k = b_k$, $b'_k = a_k$ if $k \in \mathcal{U}$ or $a'_k = a_k$, $b'_k = b_k$ otherwise (i.e., interchange a_k with b_k for each $k \in \mathcal{U}$). Then

$$
\begin{aligned}
&A_i' + B_i' - A_i - B_i \\
&= a_1 \ldots a_i \prod_{k\in\mathcal{U},k\le i} \frac{b_k}{a_k} + b_1 \ldots b_i \\
&\quad \cdot \prod_{k\in\mathcal{U},k\le i} \frac{a_k}{b_k} - a_1 \ldots a_i - b_1 \ldots b_i \\
&= a_1 \ldots a_i \prod_{k\in\mathcal{U},k\le i} \frac{1}{a_k} \Big(\prod_{k\in\mathcal{U},k\le i} b_k - \prod_{k\in\mathcal{U},k\le i} a_k \Big) \\
&\quad + b_1 \ldots b_i \prod_{k\in\mathcal{U},k\le i} \frac{1}{b_k} \Big(\prod_{k\in\mathcal{U},k\le i} a_k - \prod_{k\in\mathcal{U},k\le i} b_k \Big) \\
&= \Big(\prod_{k\in\mathcal{U},k\le i} b_k - \prod_{k\in\mathcal{U},k\le i} a_k \Big) \Big(\prod_{l\in\mathcal{U}^c,l\le i} a_l - \prod_{l\in\mathcal{U}^c,l\le i} b_l \Big) \\
&\quad > 0, \qquad (3.9)
\end{aligned}
$$

where the last inequality holds because $a_k < b_k$ for $k \in \mathcal{U}$, $a_l \geq b_l$ for $l \in \mathcal{U}^c$, and $a_l > b_l$ for at least one $l \in \mathcal{U}^c$. It follows that $S_u' > S_u$. □

We are now ready to derive an optimal static policy for $Pm\|\mathrm{E}\{\Sigma(g + wU_i)\}$, which is given in the theorem below. The policy actually says that, when the jobs have been labeled such that $f_i \geq f_j$ if $i < j$, then the first batch of jobs $1, \ldots, m$ are assigned to machines $1, \ldots, m$, respectively, each of which becomes the first to be processed on the corresponding machine; the second batch of jobs $m + 1, \ldots, m + 2m$ are assigned to machines $1, \ldots, m$, respectively, and each becomes the second to be processed; this procedure continues until all jobs have been assigned. Note that jobs $1, m + 1, 2m + 1, \ldots$, should be assigned on the same machine, jobs $2, m + 2, 2m + 2, \ldots$ should be assigned on the same machine; and so on. Note also that the time requirement to construct the policy is $O(n)$ after the jobs have been labeled as required, while to label the n jobs needs $O(n \log n)$ time. Thus, the time complexity to obtain the policy is $O(n \log n)$.

Without loss of generality, we assume, in the theorem below that the jobs have been labeled such that $f_1 \geq f_2 \geq \cdots \geq f_n$. Besides, we let q denote the integer part of n/m.

Theorem 6. *Suppose that $\alpha < w$. Then, $\zeta^* = \{(\lambda_j^*, \mathcal{S}_j^*)\}_{j=1}^m$ is an optimal static policy for $Pm\|\mathrm{E}\{\Sigma(g + wU_i)\}$, where $\mathcal{S}_j^*$ is a set of zero elements and*

$$
\lambda_j^* = \begin{cases} (j, j+m, j+2m, \ldots, j+qm), \\ \quad \text{if } j + qm \leq n, \\ (j, j+m, j+2m, \ldots, j+(q-1)m), \\ \quad \text{otherwise}, \end{cases} \qquad (3.10)
$$

for $j = 1, \ldots, m$.

Proof. As $\alpha < w$, it is easy to see that all elements of $\mathcal{S}_j^*$ for $j = 1, \ldots, m$ should be zero. Moreover, by adding dummy jobs with $f_k = 0$ after the existing jobs we can, without loss of generality, assume that under ζ^* all n_j are equal, say, to a common number u. Thus the remaining question is to show that the sequences $(\lambda_j^*)_{j=1}^m$ given by (3.10) maximize (3.3).

Let $\lambda_j = (\lambda_j(1), \lambda_j(2), \ldots, \lambda_j(u))$. Given any two machines j and j' with corresponding λ_j^* and $\lambda_{j'}^*$, let $a_i = f_{\lambda_j^*(i)}$ and $b_i = f_{\lambda_{j'}^*(i)}$ for $i = 1, \ldots, u$. If $a_{k+1} > b_k$ or $b_{k+1} > a_k$ for some $k \in \{1, \ldots, u - 1\}$, then by Lemma 2 we can increase the value of

$$
\sum_{i=1}^{u} (a_1 \ldots a_i + b_1 \ldots b_i), \qquad (3.11)
$$

by interchanging jobs between machines j and j'. This contradicts the assumption that (λ_j^*) maximize (3.3) as (3.11) is exactly the contributions to (3.3) made by the two machines. Thus we must have

$$
a_{k+1} \leq b_k \text{ and } b_{k+1} \leq a_k \quad \text{for all } k \in \{1, \ldots, u-1\}. \qquad (3.12)
$$

Moreover, Lemma 2 also tells us that (3.11) can be increased if there exist $k, l \in \{1, \ldots, u\}$ such that $a_k < b_k$ and $a_l > b_l$, hence (λ_j^*) must satisfy

$$
\text{either } a_k \leq b_k \text{ for all } k, \quad \text{or } a_k \geq b_k \text{ for all } k. \qquad (3.13)
$$

Equations (3.12)–(3.13) imply that one of the following two must hold:

$$
a_1 \geq b_1 \geq a_2 \geq b_2 \geq a_3 \geq b_3 \geq \cdots \geq a_u \geq b_u
$$

or

$$
b_1 \geq a_1 \geq b_2 \geq a_2 \geq b_3 \geq a_3 \geq \cdots \geq b_u \geq a_u,
$$

which are equivalent to

$$
f_{\lambda_j^*(1)} \geq f_{\lambda_{j'}^*(1)} \geq f_{\lambda_j^*(2)} \geq f_{\lambda_{j'}^*(2)} \geq \cdots \geq f_{\lambda_j^*(u)} \geq f_{\lambda_{j'}^*(u)} \qquad (3.14)
$$

or

$$
f_{\lambda_{j'}^*(1)} \geq f_{\lambda_j^*(1)} \geq f_{\lambda_{j'}^*(2)} \geq f_{\lambda_j^*(2)} \geq \cdots \geq f_{\lambda_{j'}^*(u)} \geq f_{\lambda_j^*(u)}. \qquad (3.15)
$$

As (3.14) or (3.15) holds for any pair of machines and $f_1 \geq \cdots \geq f_n$, it is not difficult to see that under (λ_j^*) jobs $1, \ldots, m$ must be assigned one each to the m machines and be the first to be processed; jobs $m + 1, \ldots, m + 2m$ are also assigned one each to the m machines and the second to be processed; and so on. Moreover, job $m + 1$ should be with job 1 on the same machine, job $m + 2$ with job 2, and so on. In other words, (λ_j^*) are given by (3.10). □

3.2. Dynamic Policies

Again, similar to (2.19) in Section 2.3, it can be shown that when the policy ζ is dynamic,

$$
\mathrm{ETC}(\zeta) = (\alpha - w) \sum_{j=1}^{m} \sum_{i\in\mathcal{N}_j} \mathrm{E}[e^{-\eta R_i(\zeta)}] + nw, \qquad (3.16)
$$

where $\mathcal{N}_j$ now denotes the set of jobs completed on machine j and $R_i(\zeta)$, if $i \in \mathcal{N}_j$, denotes the total uptime of machine j before job i is completed.

Clearly, if $w \leq \alpha$, then both the policy that delays the start of all jobs infinitely and the policy that processes all jobs after the deadline D has occurred, will be optimal (recall the proof for Theorem 2 and item (2) of Remark 2). Therefore, in the following we will investigate the problem with $w > \alpha$.

We first consider the case where all jobs should be assigned to the machines a priori and are prohibited from switching from one machine to another subsequently. This occurs, for example, in situations where machines are located distantly so that neither traversal of jobs between machines, nor sending jobs separately to a machine, is desirable due to the high transportation costs.

Theorem 7. *Suppose $\alpha < w$ and the allocation of jobs to machines must be performed a priori with no subsequent job switching between machines being allowed. Then, the static policy of Theorem 6 remains optimal in (i) the class of nonpreemptive dynamic policies, and (ii) the class of preemptive dynamic policies if the hazard rates $\rho_i(s)$, for all i, are nondecreasing functions of s.*

Proof. When jobs must be allocated to the machines a priori with no subsequent change being allowed, the assignment of jobs to the machine becomes a static decision. Once this is decided, the remaining question is how to schedule a fixed set of jobs on each machine. Clearly in an optimal policy, the scheduling of a fixed set of jobs on each machine must be also optimal with respect to the single-machine problem for that machine, which in either Case (i) or Case (ii) is a static policy by Theorem 3 and Corollary 1. Consequently, an optimal policy in either Case (i) or Case (ii) coincides with an optimal policy in the class of static policies, which we have proven is the policy of Theorem 6. □

We now consider the case where the restriction on job allocation is removed but the processing times P_i, for all i, are assumed to be independently, identically distributed (i.i.d.) discrete random variables with a nondecreasing hazard rate. In this case, the result of Xu et al. (1992) is applicable, which gives us Theorem 8.

Theorem 8. *Suppose $\alpha < w$ and the distribution functions of P_i are $Q_i(t) = [Q(t + t_i) - Q(t_i)]/[1 - Q(t_i)]$, where $t_1 \geqslant t_2 \geqslant \cdots \geqslant t_n$ and $Q(t)$ is a distribution function with a nondecreasing hazard rate $\rho(s)$. Then, a policy that is optimal in the class of preemptive dynamic policies should process at any time the jobs according to the nonpreemptive SEPT (shortest expected processing time) order $(1, 2, \ldots, n)$.*

Remark 4.

(1) Note that the two policies given by the two theorems above are different. Under the policy of Theorem 7, all jobs should be first assigned to the machines according to the rule of Theorem 6, which cannot be changed subsequently. Nevertheless, under the policy of Theorem 8, the assignment of jobs to the machines is not determined a priori, and whenever a machine becomes available, the job with the shortest expected processing time will be assigned to that machine for processing.

(2) Note also that the sequence $f_1 \geqslant f_2 \geqslant \cdots \geqslant f_n$ of Theorem 6 could be inconsistent with the SEPT sequence $\mathrm{E}(P_1) \leqslant \mathrm{E}(P_2) \leqslant \cdots \leqslant \mathrm{E}(P_n)$, although they look like the same at the first glance according to the definition (2.6) of f_i. Consider an example where P_1 follows a uniform distribution in [0, 1.8] while P_2 follows an exponential distribution with parameter 1. Then $E(P_1) = 0.9 < 1 = E(P_2)$, while $f_1 = E(e^{-P_1}) = 0.46 > 0.5 = f_2 = E(e^{-P_2})$, assuming $\eta = 1$.

Both Theorems 7 and 8 provide ways to construct optimal dynamic policies for the multi-machine problem, which, however, hold only when certain conditions are imposed. In general, the multi-machine dynamic scheduling problem without any restriction is very difficult. In the rest of this subsection, we shall limit our search for an optimum in the class of ODP. We shall show that an optimal solution can be constructed easily in this class.

Unlike the single-machine problem, we now have to consider the moment of time when some machines are up and some are down. Let us address first the case where preemptions are allowed. Similar to the single-machine problem (see Section 2.3), we have a set $\mathcal{J}_u(t)$ of unfinished jobs at time t. Moreover, associated with each job $i \in \mathcal{J}_u(t)$, there is a real number $T_i(t)$, which represents the realization of the processing time on that job up to t. Suppose that $f_i^u(t)$ has been computed according to (2.24) for each job $i \in \mathcal{J}_u(t)$. Clearly, if all machines are up at time t when the deadline D has not occurred, then the problem of finding an optimal ζ_o^* to process the unfinished jobs is similar to the problem studied in Section 3.1, because of the memoryless property of D. In such a case, it follows from Theorem 6 that the decision maker should pick up the m jobs with the largest values of $f_i^u(t)$, and then process them each by one of the m machines immediately. A situation needing further examination is where some machines are down at the moment, which are expected to be up some time later. It might be desirable that, at a present time t, a job should be assigned to a "down" machine to wait until it becomes available. To deal with this problem, let us investigate a simple case first, in which there are two machines, with machine 1 being up and machine 2 having been down for a period of time $r_2 > 0$. Assume that a set of jobs with known values of $f_i^u(t)$ are to be processed on these two machines.

It can be seen that on machine 2, (2.2) becomes $Z_{2,0} \geqslant r_2$. Accordingly, if job i is to be processed by machine 2, then (2.9) becomes (conditional on the available information at time t that machine 2 has already been down for a period of r_2):

$$\begin{aligned}&\Pr(D > C_i(\zeta) | R_i(\zeta) = x)\\ &\quad = e^{-\delta x}\Pr(D > Z_{2,0} - r_2 | Z_{2,0} \geqslant r_2)\\ &\qquad \cdot \sum_{k=0}^{\infty} \Pr(D > Z_{2,1} + \cdots + Z_{2,k})\Pr(N_2(x-) = k).\end{aligned}$$

Thus, following a similar argument as in Section 2.1, after some development one may show that the objective function for this two-machine problem becomes:

$$\mathrm{ETC}(\zeta) = (\alpha - w)\Big\{ \sum_{i\in\mathcal{N}_1} \prod_{k\in\mathcal{B}_i(\lambda_1)} e^{-\eta s_k} f_k^u(t) + \gamma_2 \sum_{i\in\mathcal{N}_2} \prod_{k\in\mathcal{B}_i(\lambda_2)} e^{-\eta s_k} f_k^u(t)\Big\} + (n_1 + n_2)w, \quad (3.17)$$

where $\gamma_2 = \Pr(D > Z - r_2 | Z \geq r_2)$ (see Theorem 1 for the definition of Z). Note that $0 \leq \gamma_2 < 1$. Since $\alpha < w$, it is easy to see that minimizing (3.17) is equivalent to choosing $s_i = 0$ for all i and then determining two optimal sequences λ_1^* and λ_2^* so as to maximize:

$$\mathrm{ETC}''(\zeta) = \sum_{i\in\mathcal{N}_1} \prod_{k\in\mathcal{B}_i(\lambda_1)} f_k^u(t) + \gamma_2 \sum_{i\in\mathcal{N}_2} \prod_{k\in\mathcal{B}_i(\lambda_2)} f_k^u(t). \quad (3.18)$$

The following result is related to the maximization of (3.18) above.

Lemma 3. *Let $S_u = A_1 + \cdots + A_u + \gamma_2(B_1 + \cdots + B_u)$, where $\gamma_2 < 1$ is a constant and A_i and B_i are as defined in Lemma 2. If $a_k < b_k$ for some $k \in \{1, \ldots, u\}$, then there exist A_i' and B_i' (see (3.5)) such that $S_u' = A_1' + \cdots + A_u' + \gamma_2(B_1' + \cdots + B_u)' > S_u$.*

Proof. Let $\mathcal{U} = \{k : a_k < b_k\}$. Similar to (3.9), it is easy to verify that

$$A_i' + \gamma_2 B_i' - A_i - \gamma_2 B_i = \Big(\prod_{k\in\mathcal{U},k\leq i} b_k - \prod_{k\in\mathcal{U},k\leq i} a_k\Big) \cdot \Big(\prod_{k\notin\mathcal{U},k\leq i} a_k - \gamma_2 \prod_{k\notin\mathcal{U},k\leq i} b_k\Big), \quad (3.19)$$

which is positive since $\gamma_2 < 1$ and $\mathcal{U}$ is nonempty by the assumption. The present lemma is then proven following a similar argument as in the proof of Lemma 2. □

Lemma 3 actually implies that, if $\lambda_1^* = \{\lambda_1^*(1), \lambda_1^*(2), \ldots\}$ and $\lambda_2^* = \{\lambda_2^*(1), \lambda_2^*(2), \ldots\}$ are the job sequences on the two machines, respectively, then $\mathrm{ETC}''(\zeta)$ of (3.18) is not maximized unless $f^u_{\lambda_1^*(1)}(t) \geq f^u_{\lambda_2^*(1)}(t)$, $f^u_{\lambda_1^*(2)}(t) \geq f^u_{\lambda_2^*(2)}(t)$, and so on. (Note that by adding dummy jobs with $f_k^u(t) = 0$ after the existing jobs on machine 2 we may assume that the two sequences contain the same number of jobs.) Moreover, Theorem 2 indicates that a job sequence on a machine is not optimal if it does not sequence the jobs assigned to this machine in nonincreasing order of $f_i^u(t)$. This together with $f^u_{\lambda_1^*(1)}(t) \geq f^u_{\lambda_2^*(1)}(t)$ leads to the following result.

Corollary 4. *At any time t before the occurrence of D, $\mathrm{ETC}''(\zeta)$ of (3.18) does not attain its maximum if the job with the largest $f_i^u(t)$ is not sequenced as the first job on machine 1.*

Corollary 4 above has been sufficient for us to determine an optimal policy ζ_o^* in the class of ODP for the multi-machine problem.

Theorem 9. *Suppose that $\alpha < w$ and preemptions are allowed. Then, a policy ζ_o^* for $Pm\|\mathrm{E}\{\Sigma(g + wU_i)\}$ will process the jobs according to the following rules*:

(a) *At any time t before the occurrence of D, assign the jobs in $\mathcal{J}_u(t)$ with largest values of $f_i^u(t)$ to the machines which are up at this time t. One machine should receive and process one job immediately with no inserted idle time.*

(b) *After the occurrence of D, all unfinished jobs may be processed on any machine in any order, and preemptions are not necessary.*

Proof. According to Corollary 4, a decision is not optimal if, at time t, it assigns job i to a machine that is down and job j to a machine that is up, if $f_i^u(t) > f_j^u(t)$. This together with the argument that no inserted idle times are optimal before the occurrence of D proves Part (a). The proof for part (b) is straightforward. □

We now turn to the case where preemptions are forbidden. In this case, we have to consider such a moment t when some machines have just finished their current jobs and are ready to accept new jobs, some machines are up and are processing, and some machines are down at the moment. Let $\mathcal{M}_f$, $\mathcal{M}_p$, and $\mathcal{M}_d$ denote the three sets of machines, respectively. As preemptions are not allowed, the ready time for accepting a new job of a machine in $\mathcal{M}_p$ is random since the remaining processing time of the job it is processing is random. This is similar to introducing a factor $0 \leq \xi < 1$ like γ_2 of (3.18) on this machine. Thus, based on the analysis above, we have Theorem 10.

Theorem 10. *Suppose that $\alpha < w$ and preemptions are not allowed. Then, a policy ζ_o^* for $Pm\|\mathrm{E}\{\Sigma(g + wU_i)\}$ will process the jobs according to the following rules*:

(a) *At any time t before the occurrence of D, assign the unprocessed jobs with largest values of f_i to the machines in $\mathcal{M}_f$, with one job being processed by one machine without any inserted idle times.*

(b) *After the occurrence of D, all unprocessed jobs may be assigned to any machines in any order.*

Remark 5.

(1) Recall that in the single-machine case, the optimal static policy remains an optimal dynamic policy if preemptions are not allowed (see Theorem 3). Such a conclusion, however, does not hold in the multi-machine case even though the optimum is to be sought from the class of ODP. This is because the job sequence on each machine may change dynamically due to the re-allocation of jobs to different machines.

(2) Recall also Corollary 3. The conclusion there for the single-machine case is not applicable in the multi-machine case either. This is also because of the possible re-allocation of jobs to different machines. Even the computation of $f_i^u(t)$ cannot be saved in the multi-machine case, even if the processing times satisfy the condition of Lemma 1. Consider the scenario where two machines are in the down status at a same time, and the two jobs that they were processing, respectively, before their breakdowns can be switched to another available machine. In this case, the values of $f_i^u(t)$ (but not f_i) of the two jobs

will determine which job is more important and should get immediate processing by the available machine, according to Theorem 9.

3.3. Generalizations

The results derived in Section 3.1 on static policies can be generalized to a more general case with the following compatibility condition:

$$f_i > f_j \Rightarrow (w_i - \alpha_i) f_i \geq (w_j - \alpha_j) f_j, \quad \forall\, i, j \in \mathcal{J}. \tag{3.20}$$

Notice that various types of compatibility conditions of similar nature have been widely used in the literature; see Lawler (1977), Kise et al. (1978 and 1982), Weiss and Pinedo (1980), Pinedo (1983), Kampke (1989), Emmons and Pinedo (1990), Lee et al. (1991), Cai (1995), etc.

To be more precise, the main result is summarized below. The proof is omitted here as it can be established following similar arguments as in Section 3.1. Similar to Theorem 6, in Theorem 11 we assume, without loss of generality, that $\mathcal{J} = \{1, 2, \ldots, n_I\}$ and $f_i \geq f_j$ if $i < j$ for $i, j \in \mathcal{J}$.

Theorem 11. *Under the compatibility condition* (3.20), *an optimal static policy for the stochastic problem* $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *will process the jobs according to the following rules*:

(a) *Jobs in* $\mathcal{J}$ *are sequenced according to* (3.10) (*with n being replaced by* n_I) *and processed without any inserted idle times*.

(b) *All jobs with* $\alpha_i \geq w_i$ *may be processed on any machine in any order, starting as soon as D has occurred or the last job with* $\alpha_i < w_i$ *has been completed, whichever is the later*.

Remark 6. It can be verified that in the following situations the compatibility condition (3.20) is satisfied:

(a) $w_i \equiv w$ and $g_i(\cdot) \equiv g(\cdot)$, $\forall i \in \mathcal{J}$;

(b) All processing times P_i are i.i.d., $\forall i \in \mathcal{J}$;

(c) $f_i > f_j \Rightarrow w_i \geq w_j$ when $\alpha_i \equiv \alpha$, $\forall i, j \in \mathcal{J}$;

(d) P_i can be stochastically ordered and $P_i <_{s.t.} P_j \Rightarrow w_i - \alpha_i \geq w_j - \alpha_j$, $\forall i, j \in \mathcal{J}$, where a random variable X is said to be stochastically less than a random variable Y, denoted by $X <_{s.t.} Y$, if $\Pr(X > a) \leq \Pr(Y > a)$ for all a with strict inequality holding for some a;

(e) P_i can be stochastically ordered and $P_i <_{s.t.} P_j \Rightarrow w_i \geq w_j$ when $\alpha_i \equiv \alpha$, $\forall i, j \in \mathcal{J}$.

Case (a) is what we studied in Section 3.1 above. Problems with i.i.d. processing times are frequently addressed in the literature; see, for example, Boxma and Forst (1986), Buzacott and Shanthikumar (1993), Coffman et al. (1993), Emmons and Pinedo (1990), Xu et al. (1992), etc. Emmons and Pinedo have indicated that for parallel machine problems to minimize the expected weighted number of tardy jobs, the assumption of i.i.d. processing times is generally necessary to obtain analytical policies. Case (c) deals with the situation where the earliness cost is common for all jobs. Case (d) above is intuitively understandable, since if the processing times of a job i is "smaller" than that of another job j, and the difference of job i between its tardiness cost and its earliness cost is bigger than that of job j, then naturally job i should be processed before job j as this will reduce the risk of incurring more cost. Case (e) is a special case of case (d).

As far as dynamic policies are concerned, it is obvious that Theorem 7 can also be generalized under the compatibility condition (3.20). This leads to Theorem 12.

Theorem 12. *Suppose that* (3.20) *is satisfied and allocation of jobs to machines must be performed a priori with subsequent job switching between machines being prohibited. Then, the static policy of Theorem* 11 *remains optimal for the stochastic problem* $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$, *in* (*i*) *the class of nonpreemptive dynamic policies, and* (*ii*) *the class of preemptive dynamic policies if the hazard rates* $\rho_i(s)$, *for all* $i \in \mathcal{J}$, *are nondecreasing functions of s*.

It is unclear whether Theorem 8 can be generalized to the weighted case. Nevertheless, under a stronger compatibility condition, Theorems 9 and 10 can be generalized as follows (the proof is again omitted here as it can be obtained following similar arguments as in Section 3.2).

Theorem 13. *Suppose that for all i and j in* $\mathcal{J}$, $f_i^u(t) > f_j^u(t) \Rightarrow (w_i - \alpha_i) f_i^u(t) \geq (w_j - \alpha_j) f_j^u(t)$ *for all* $t \geq 0$. *Then, a policy that is optimal in the class of ODP for* $Pm\|\mathrm{E}\{\Sigma(g_i + w_iU_i)\}$ *will process the jobs according to the following rules*:

(a) *Jobs in* $\mathcal{J}$ *should be processed according to Rule* (*a*) *of Theorem* 9 *if preemptions are allowed, or Rule* (*a*) *of Theorem* 10 *if preemptions are forbidden*;

(b) *All jobs with* $\alpha_i \geq w_i$ *may be processed on any machine in any order, starting as soon as D has occurred or the last job with* $\alpha_i < w_i$ *has been completed, whichever is the later, no matter whether preemptions are allowed or forbidden*.

4. CONCLUDING REMARKS

We studied a stochastic scheduling problem with the following main features: the processing times of the jobs may follow any arbitrary distributions, the machines are subject to stochastic breakdowns, the deadline is a random variable that models the situations where jobs have to be scheduled carefully against an event that happens with uncertainty. Moreover, the performance measure of the problem is to minimize the expected total cost for earliness and tardiness. This differs from the majority of the prior stochastic scheduling literature. In our performance measure, the earliness cost is a general, arbitrary function. This provides a practical and flexible model for decision-making. Although there have been some studies in stochastic scheduling that involve other earliness/tardiness measures, it appears that none of them has addressed issues like multiple machines subject to random breakdowns, dynamic policies, preemptions, etc., as we investigate in this article. In particular, it seems that our work is the first one in stochastic earliness/tardiness scheduling in which optimal policies are found to be obtainable analytically

even though the corresponding deterministic counterparts of the problem have been known to be NP-hard.

The work of this article may lead to numerous topics for further studies. These include generalizations to problems with:

- Dynamic arrivals of the jobs.
- Nonidentical parallel machines.
- A set-up cost before starting a job. This is particularly important in cases where job preemptions are allowed.
- An individual deadline D_i for each job i, $i \in \mathcal{N}$. Here we should point out that all the results we have obtained in this article can actually be generalized to this case as long as all D_i follow a same exponential distribution. (Note that in this case we can still get Theorem 1 by following the same procedure as in Section 2.1.)
- General w_i and $g_i(\cdot)$ for each $i \in \mathcal{N}$ in the multi-machine case. Note that in this article we still need the compatibility conditions of Section 3.3. The policies obtained under the compatibility conditions can, of course, be used in the general case without these conditions if one is satisfied with approximate solutions.

We are at present working on the multi-machine problem with general w_i and $g_i(\cdot)$. The results will be reported in a separate paper.

ACKNOWLEDGMENTS

The authors wish to thank the associate editor and the anonymous referees for numerous suggestions and comments that greatly improved the paper. This research was supported in part by The Hong Kong Government Industry Department under Grant No. AF/148/94, and by the Research Grants Council of Hong Kong under Earmarked Grant No. CUHK 356/96E.

REFERENCES

Allahverdi, A., J. Mittenthal. 1994. Scheduling on *M* parallel machines subject to random breakdowns to minimize expected mean flow time. *Naval Res. Logistics* **41** 677–682.

Baker, K. R., G. D. Scudder. 1990. Sequencing with earliness and tardiness penalties: a review. *Opns. Res.* **38** 22–36.

Balut, S. J. 1973. Scheduling to minimize the number of late jobs when set-up and processing times are uncertain. *Management Sci.* **19** 1283–1288.

Bayard, D. S. 1987. Proof of quasi-adaptivity for the *m*-measurement feedback class of stochastic control policies. *IEEE Trans. on Automatic Control* **32** 447–451.

Bertsekas, D. P. 1976. *Dynamic Programming and Stochastic Control*. Academic Press, New York.

Birge, J., J. B. G. Frenk, J. Mittenthal, A. H. G. Rinnooy Kan. 1990. Single-machine scheduling subject to stochastic breakdowns. *Naval Res. Logistics* **37** 661–677.

Blau, R. A. 1973. *N*-job, one machine sequencing problems under uncertainty. *Management Sci.* **20** 101–109.

Blazewicz, J., K. H. Ecker, G. Schmidt, J. Weglarz. 1994. *Scheduling in Computer and Manufacturing Systems*, 2nd Edition. Springer-Verlag, Berlin.

Boxma, O. J., F. G. Forst. 1986. Minimizing the expected weighted number of tardy jobs in stochastic flow shops. *Oper. Res. Letters* **5** 119–126.

Buzacott, J. A., J. G. Shanthikumar. 1993. *Stochastic Models of Manufacturing Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Cai, X. 1995. Minimization of agreeably weighted variance in single machine systems. *European J. Oper. Res.* **85** 576–592.

——. 1996. V-shape property for job sequences that minimize the expected completion time variance. *European J. Oper. Res.* **91** 118–123.

——, S. Zhou. 1997. Scheduling stochastic jobs with asymmetric earliness and tardiness penalties. *Naval Res. Logistics* **44** 531–557.

Chakravarthy, S. 1986. A single machine scheduling problem with random processing times. *Naval Res. Logistics Quarterly* **33** 391–397.

Cheng, T. C. E. 1986. Optimal due-date assignment for a single machine sequencing problem with random processing times. *International J. Systems Sci.* **17** 1139–1144.

——. 1987. Optimal due-date determination and sequencing with random processing times. *Math. Modelling* **9** 573–576.

Coffman, E. G. Jr., L. Flatto, P. E. Wright. 1993. Optimal stochastic allocation of machines under waiting-time constraints. *SIAM J. Comput.* **22** 332–348.

De, P., J. B. Ghosh, C. E. Wells. 1991. On the minimization of the weighted number of tardy jobs with random processing times and deadline. *Computers & Oper. Res.* **18** 457–463.

Derman, C., G. Lieberman, S. Ross. 1978. A renewal decision problem. *Management Sci.* **24** 554–561.

Emmons, H., M. Pinedo. 1990. Scheduling stochastic jobs with due dates on parallel machines. *European J. Oper. Res.* **47** 49–55.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, New York.

Gittins, J. C. 1989. *Multi-Armed Bandit Allocation Indices*. John Wiley & Sons, Chichester, England.

Glazebrook, K. D. 1984. Scheduling stochastic jobs on a single machine subject to breakdowns. *Naval Res. Logistics Quarterly* **31** 251–264.

Kahlbacher, H. G., T. C. E. Cheng. 1993. Parallel machine scheduling to minimize costs for earliness and number of tardy jobs. *Discrete Appl. Math.* **47** 139–164.

Kampke, T. 1989. Optimal scheduling of jobs with exponential service times on identical parallel processors. *Oper. Res.* **37** 126–133.

Karp, R. M. 1972. Reducibility among combinatorial problems. *Complexity of Computer Computations*. R. E. Miller, J. W. Thatcher, eds. Plenum Press, New York.

Katoh, N., T. Ibaraki. 1983. A polynomial time algorithm for a chance-constrained single machine scheduling problem. *Oper. Res. Letters* **2** 62–65.

Kise, H., T. Ibaraki. 1983. On Balut's algorithm and NP-completeness for a chance-constrained scheduling problem. *Management Sci.* **29** 384–388.

——, ——, H. Mine. 1978. A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.* **26** 121–126.

——, A. Shiomi, M. Uno, D. Chao. 1982. An efficient algorithm for a chance-constrained scheduling problem. *J. Oper. Res. Society of Japan* **25** 193–203.

Lee, C.-Y., S. L. Danusaputro, C.-S. Lin. 1991. Minimizing weighted number of tardy jobs and weighted earliness-tardiness penalties about a common due date. *Computers & Oper. Res.* **18** 379–389.

Lawler, E. L. 1977. A pseudopolynomial algorithm for sequencing jobs to minimize total tardiness. *Ann. Discrete Math.* **1** 331–342.

Mayne, D. Q., H. Michalska. 1990. Receding horizon control of nonlinear systems. *IEEE Trans. Automatic Control* **35** 814–824.

Mittenthal, J., M. Raghavachari. 1993. Stochastic single machine scheduling with quadratic early-tardy penalties. *Oper. Res.* **41** 786–796.

Pinedo, M. 1983. Stochastic scheduling with release dates and due dates. *Oper. Res.* **31** 559–572.

——, S. H. Ross. 1980. Scheduling jobs subject to nonhomogeneous Poisson shocks. *Management Sci.* **26** 1250–1257.

Sarin, S. C., E. Erel, G. Steiner. 1991. Sequencing jobs on a single machine with a common due date and stochastic processing times. *European J. Oper. Res.* **51** 287–302.

Soroush, H. M., L. D. Fredendall. 1994. The stochastic single machine scheduling problem with earliness and tardiness costs. *European J. Oper. Res.* **77** 287–302.

Tse, E., M. Athans. 1972. Adaptive stochastic control for a class of linear systems. *IEEE Trans. Automatic Control* **17** 38–51.

Vani, V., M. Raghavachari. 1987. Deterministic and random single machine sequencing with variance minimization. *Oper. Res.* **35** 111–120.

Weber, R. 1992. On the Gittins index for multiarmed bandits. *Ann. Appl. Probab.* **2** 1024–1033.

Weiss, G., M. Pinedo. 1980. Scheduling tasks with exponential services times on non-identical processors to minimize various cost functions. *J. Appl. Probab.* **17** 187–202.

Xu, S. H., S. P. R. Kumar, P. B. Mirchandani. 1992. Scheduling stochastic jobs with increasing hazard rate on identical parallel machines. *Computers & Oper. Res.* **19** 535–544.