IIE Transactions

ISSN: 0740-817X (Print) 1545-8830 (Online) Journal homepage: http://www.tandfonline.com/loi/uiie20

Optimization of schedule robustness and stability under random machine breakdowns and processing time variability

Selcuk Goren & Ihsan Sabuncuoglu

IIE Transactions (2010) **42**, 203–220

ISSN: 0740-817X print / 1545-8830 online
DOI: 10.1080/07408170903171035

Optimization of schedule robustness and stability under random machine breakdowns and processing time variability

SELCUK GOREN and IHSAN SABUNCUOGLU*

Department of Industrial Engineering, Bilkent University, Ankara, Turkey
E-mail: sabun@bilkent.edu.tr

Received November 2006 and accepted June 2009

In practice, scheduling systems are subject to considerable uncertainty in highly dynamic operating environments. The ability to cope with uncertainty in the scheduling process is becoming an increasingly important issue. This paper takes a proactive scheduling approach to study scheduling problems with two sources of uncertainty: processing time variability and machine breakdowns. Two robustness (expected total flow time and expected total tardiness) and three stability (the sum of the squared and absolute differences of the job completion times and the sum of the variances of the realized completion times) measures are defined. Special cases for which the measures can be easily optimized are identified. A dominance rule and two lower bounds for one of the robustness measures are developed and subseqently used in a branch-and-bound algorithm to solve the problem exactly. A beam search heuristic is also proposed to solve large problems for all five measures. The computational results show that the beam search heuristic is capable of generating robust schedules with little average deviation from the optimal objective function value (obtained via the branch-and-bound algorithm) and it performs significantly better than a number of heuristics available in the literature for all five measures.

Keywords: Proactive scheduling, robustness, stability

1. Introduction

Scheduling is a decision-making process that is concerned with the allocation of limited resources (machines, material-handling equipment, operators, tools, fixtures, etc.) to competing tasks (operations of jobs) over time, with the goal of optimizing one or more objectives. The output of this decision process is time/machine/operation assignments. In the scheduling literature, the objective is generally to minimize functions such as makespan, tardiness, flow time, etc.

In practice, scheduling systems operate in dynamic and uncertain environments in which random interruptions prevent the execution of a schedule exactly as it is developed. Examples of such disruptions are machine breakdowns, rush orders, order cancellations, due date changes, etc. Variation in processing times and other stochastic events further increase the variability in the system, which in turn deteriorate the scheduling performance.

Even though actual scheduling problems in real life are dynamic and stochastic, most of the existing literature addresses static and deterministic versions. However, even these simplified problems (with deterministic and static assumptions) are NP-hard or analytically intractable.

The uncertainties and dynamic nature of the real-world scheduling process can be seen as the major source of the gap between scheduling theory and practice. In the literature, several studies have been conducted to close this gap. In the early works, researchers employ a rolling-horizon scheme to cope with the dynamic nature of scheduling environments, where the problem is successively solved using static algorithms for different time windows (Nelson *et al.*, 1977). The stochastic nature of scheduling has also been investigated in the literature. In these studies, uncertainty in job processing times, release times, or due dates is modeled by probability distribution functions and formal probability theory is used to make inferences (Pinedo, 2002, Chapters 9–13). In the last two decades researchers have also proposed approaches including online scheduling, dynamic scheduling, and real-time scheduling. Recently, two approaches to coping with uncertainty in the scheduling process have gained significant research interest: *reactive* and *proactive* scheduling. The objective in reactive scheduling is to revise schedules as unexpected events (disruptions) occur. On the other hand, proactive scheduling takes future disruptions into account while generating schedules.

The challenge of addressing the dynamic and stochastic nature of the scheduling process also affects the

*Corresponding author

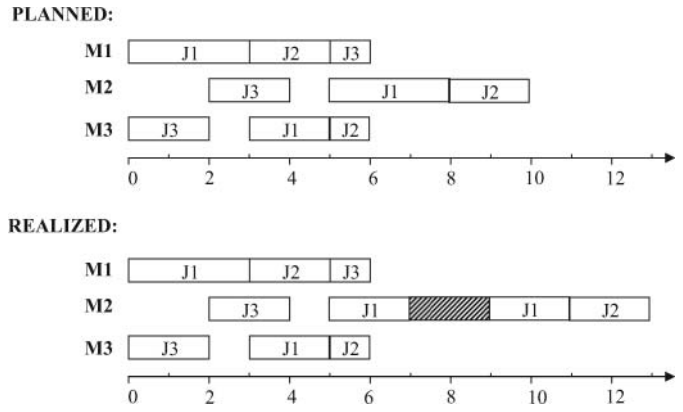

Fig. 1. An initial schedule and its realization for J3$||C_{\max}$.

performance measure of choice. Although performance measures such as makespan, flow time, or tardiness have often been preferred in practice, in the recent literature two new measures are brought to the attention of practitioners: *robustness and stability*. These measures are particularly used in environments where uncertainty is a major issue.

Uncertainty has two kinds of major negative impacts on initial schedules. First, it degrades schedule performance. This effect is the topic of *robustness*. A schedule whose performance does not deteriorate in the face of disruptions is called *robust*. In other words, the performance of a robust schedule is expected to be insensitive to disruptions. In general, the performance of the realized schedule is the main concern of practitioners rather than the planned or estimated performance of the initial schedule. Hence, optimizing the former may be more appropriate than optimizing the latter and robustness is a practical performance measure. Second, unforeseen disruptions cause variability. This effect is the topic of *stability.* A schedule whose realization does not deviate from the original schedule in the face of disruptions is called *stable*. A schedule serves as a master plan for other shop floor activities in addition to production, such as determining delivery dates, release times, and planning requirements for secondary resources such as tools, fixtures, etc. Any deviation from the production schedule can disrupt these secondary activities and increase system nervousness. Thus, stability is also an important measure in practice.

Robustness and stability can be illustrated with the help of Fig. 1. The top Gantt chart in the figure depicts a possible initial schedule for a job shop environment with three jobs and three machines subject to random breakdowns. The bottom Gantt chart shows a possible realization of the initial schedule. The shaded area on the realized schedule of machine 2 between times 7 and 9 represents a breakdown. Assume that the performance measure of interest is the maximum completion time ($C_{\max}$). From the robustness viewpoint, the scheduler should be concerned with the performance of the realized schedule ($C_{\max} = 13$ in the example) rather than the performance of the initial schedule ($C_{\max} = 10$ in the example). Hence, he/she optimizes a measure (*robustness measure*) that is defined on the *realized* schedule. Another way to look at this is to minimize the performance deviation between the initial and the realized schedules ($\Delta C_{\max} = 13 - 10 = 3$ in the example). Observe that the operation of job 1 on machine 2 completes later than planned. Similarly, while the operation of job 2 on that machine is planned to be processed between times 8 and 10, it is actually processed between times 11 and 13 because of the breakdown. From the stability viewpoint, such deviations from the initial schedule (i.e., the master plan) should be minimized. Hence, the scheduler optimizes a measure (*stability measure*) defined in terms of the deviations between the initial and the realized schedules.

The reactive and proactive scheduling approaches and these two new performance measures (robustness and stability) are discussed in more detail in Sabuncuoglu and Goren (2009).

In this study, we consider the single-machine scheduling problem with random processing times and machine breakdowns. When a breakdown occurs, the machine is unavailable until it is repaired. The times for repair are also random and independent of each other and of the breakdown process. A job preempted due to a breakdown is processed for its remaining processing time (i.e., preempt-resume policy is assumed). No other preemptions are allowed. As stated before, we take a proactive point of view and define several robustness and stability measures.

Although there are some studies which measure robustness as a minimax regret (e.g., Daniels and Kouvelis (1995), the majority of recent studies on robustness involve expected realized performance. The expected realized performance can be the robustness measure by itself (e.g., Wu *et al.* (1999) or can be a part of it (e.g., Leon *et al.* (1994)). In this study, we use the expected realized performance measure as the robustness measure. We consider two performance measures: expected total flow time ($RM1$) and expected total tardiness ($RM2$).

The most frequent way to measure the deviation between the initial and the realized schedules (stability) is to compare their job completion times (Wu *et al.*, 1993; Mehta and Uzsoy, 1998). We use two stability measures based on this comparison: the sum of the squared differences ($SM1$) and the sum of the absolute differences ($SM3$). We also use the sum of the variances of the realized completion times as another stability measure ($SM2$). The rationale behind this and how it corresponds to the difference between the initial and the realized schedules is explained in Section 3.4.1. Note that all these stability measures can be trivially minimized by inserting large blocks of idle times between jobs in the initial schedule. In this study, however, we confine ourselves to the class of non-delay schedules.

We also derive optimality conditions and propose a proactive branch-and-bound algorithm, which uses a stochastic dominance rule, for minimizing the expected

total tardiness (*RM*2). We consider a single-machine environment because of its simplicity and the possible extendibility of its results to more realistic multi-machine environments.

The rest of this paper is organized as follows. In Section 2, we review the existing proactive studies in the literature. In Section 3, we approach the proactive scheduling problem in a single-machine environment using probability theory. In Section 4, we present a branch-and-bound algorithm that utilizes insights gained in Section 3 to minimize the expected total tardiness in a single-machine environment with variable processing times. We present a beam search algorithm that can handle other performance measures and machine breakdowns in Section 5. In Section 6, we assess the performance of the proposed algorithms with computational experiments. Finally, we make concluding remarks and discuss future research directions in Section 7.

2. Literature review

Although this study is on schedule robustness and stability, the literature on scheduling with unreliable machines is relevant. We first review a few studies in this area.

Adiri *et al.* (1989) consider the problem of minimizing total flow time in a single-machine environment subject to random breakdowns. In contrast with our study, only one machine breakdown occurs and a preempt-repeat policy is assumed. The authors show that if the distribution function of the time to breakdown is concave, then the Shortest Processing Time (SPT) first rule stochastically minimizes the flow time. For the case of multiple breakdowns, it is proven that SPT minimizes the expected flow time when the times to breakdown are exponentially distributed. The authors show that the problem is NP-hard even when the time for the single breakdown is known in advance and the processing times of the jobs are deterministic.

In a later study, Adiri *et al.* (1991) consider the single-machine scheduling problem with deterministic processing times and due dates subject to a single random breakdown. The authors develop policies to minimize the number of tardy jobs stochastically, working under certain assumptions for both preempt-resume and preempt-repeat policies.

Similar to our study, Li and Glazebrook (1998) consider the single-machine scheduling problem with random processing times and multiple machine breakdowns with a preempt-resume policy. The objective is to minimize a weighted sum of an increasing function of the completion times in expectation. The authors develop a dominance rule based on pairwise interchanges of adjacent jobs. The rule is also relaxed to allow uptimes to be distributed as a mixture of exponentials and according to a gamma distribution. The dominance rule, however, cannot be applied to due-date-related measures, which are not functions of completion times only. We develop a similar dominance rule based on pairwise interchanges of jobs (not necessarily adjacent) for the total tardiness measure for the case of no machine breakdowns.

Li *et al.* (1998) consider the same problem under Erlang uptime distribution. All jobs are assumed to have a common exponentially distributed due date (compared with deterministic but different due dates in our study). The authors develop dominance rules based on pairwise interchanges of adjacent jobs in order to minimize the weighted number of tardy jobs, weighted flow times and weighted sum of job delays.

Leung and Pinedo (2004) study the preemptive parallel-machine scheduling problem with random breakdowns and deterministic processing times and due dates. The authors develop conditions on the number of available machines $m(t)$ that minimize total completion time, makespan or maximum lateness. The authors also analyze cases with deadlines and precedence constraints.

We refer interested readers to Pinedo (2002) to see a concise summary of stochastic scheduling results. Next, we review the studies in the literature that explicitly address robustness or stability of schedules. These studies can be divided into two parts—those that model uncertainty by probability density functions and those that hedge against the worst contingency that may arise without considering any specific probability distribution. The latter is known as the *robustness approach* in the literature. In both approaches, the source of uncertainty is either the variability of task processing times or machine availability (the machines are subject to a breakdown/repair process).

Leon *et al.* (1994) is an example of the first approach. They consider the job shop scheduling problem with machine breakdowns. The objective is to construct a robust initial schedule. The robustness measure for a schedule is calculated as a convex combination of the expected makespan of the realized schedule and the expected deviation from the initial deterministic makespan. In a job shop environment with multiple machine failures, however, calculating this measure analytically is intractable. They develop a surrogate measure and minimize that measure instead. The results indicate that the proposed algorithm outperforms the classical algorithms that focus on minimizing makespan only.

Wu *et al.* (1999) propose a graph-theoretic decomposition for the job shop scheduling problem to achieve schedule robustness. Expected average weighted tardiness is used as the robustness measure. The authors use a graph representation of this problem, in which conjunctive arcs represent precedence constraints and disjunctive arcs join operations competing for the same resource. They propose a branch-and-bound algorithm that processes disjunctive arcs and changes *some* of them into conjunctive arcs. This effectively fixes some of the scheduling decisions. The remaining scheduling decisions are made dynamically by applying the Apparent Tardiness Cost (ATC) heuristic (Vepsalainen and Morton, 1987). Their

computational experiments indicate that this scheme displays better robustness performance under a wide range of disturbance levels (various levels of processing time variability) compared to traditional offline and online methods.

There are also studies that model uncertainty with probability density functions with the aim of generating stable schedules. For example, Wu *et al.* (1993) study the single-machine rescheduling problem under machine disruptions. They reschedule the jobs in response to each machine failure so that a minimum makespan is obtained with high schedule stability (the measure they use is similar to *SM*3). Since the problem is NP-hard even without stability considerations, they use a pairwise swapping heuristic and a genetic algorithm to generate a list of non-dominated schedules. Their computational results show that the stability of the schedules could be improved significantly with little sacrifice in makespan.

Mehta and Uzsoy (1998, 1999) generate initial stable schedules under random machine breakdowns. Their objective is to generate an initial schedule with minimal deviation (i.e., *SM*3) while keeping shop floor performance degradation at an acceptable level. The specific problem they study in the first paper is the single-machine scheduling problem where jobs have unequal ready times and random machine breakdowns are present. In the second paper, they study the job shop scheduling problem with random machine breakdowns. In both studies, they use maximum lateness as the shop floor performance measure. Unlike Wu *et al.* (1993), they consider the minimization of the deviation between the initial and the realized schedule while generating an initial schedule, not when rescheduling after a breakdown. The authors offer a two-stage approach. In the first stage, a job sequence that will minimize the maximum lateness is determined. In the second stage, they insert idle times into the sequence. Their computational results indicate that stability can be easily improved while slightly increasing maximum lateness.

O'Donovan *et al.* (1999) combine the reactive and the proactive approaches and examine the scheduling/rescheduling policy using stability and efficiency measures in a single-machine environment. Schedule efficiency is measured by total tardiness (*RM*2). Stability is measured by absolute completion time deviations from the initial schedule (*SM*3). The system under study has non-zero job ready times and random machine breakdowns. This study is similar to the one by Mehta and Uzsoy (1999) except that total tardiness is used instead of maximum lateness. They consider pure ATC and ATC with inserted idle times for initial schedule generation. Rescheduling alternatives are ATC, a modified ATC (which calculates the slack of a job based on its predicted completion time, taking inserted idle times into account), and right-shift scheduling. Their results indicate that ATC with inserted idle times for an initial schedule and the modified ATC for rescheduling are the best for stability.

For the robustness approach, we refer the reader to Kouvelis and Yu (1996), who apply this method to various problems such as linear programming, assignment problem, shortest path problem, etc., as well as scheduling. An example of such an approach in the machine scheduling context is the study of Daniels and Kouvelis (1995). They generate initial robust schedules to hedge against processing time variability in a single-machine environment. The authors propose a scenario-based representation and analysis of uncertainty rather than using stochastic models. They use a policy that finds the schedule whose performance degradation in its worst-case scenario is the least among all feasible schedules (i.e., minimax regret strategy in decision theory). The authors study a single-machine problem where the performance measure is total flow time, and the source of uncertainty is processing time variability. The authors prove that a properly selected finite set of scenarios is enough to determine the worst-case absolute deviation of a given sequence and construct a procedure that calculates the worst-case evaluation in polynomial time. They develop a branch-and-bound algorithm and two $O(n \log n)$ surrogate relaxation heuristics that utilize this procedure to generate robust schedules. The authors compare their solutions to the SEPT (Shortest Expected Processing Time) solution, which is used in practice to generate an optimal sequence of jobs. They observe that SEPT performs poorly in terms of robustness.

Such a minimax regret approach to robustness may be more appropriate than the more frequently used expected performance measure approach if the distributions that capture the uncertainty are unknown or imprecise. Additionally, in many cases a stochastic approach that models the uncertainty with probability density functions assumes distributional independence to improve analytical tractability. If such an assumption is invalid (i.e., strong correlations exist among the probability distributions), a minimax regret approach may be more suitable to employ. Finally, if the scheduling decisions are evaluated *ex post* (as if all the relevant information had been known in advance of scheduling), a decision maker may be inclined to reduce the difference between the realized performance and the optimal performance that could have been achieved (i.e., minimize regret), rather than the average performance (Daniels and Kouvelis, 1995).

Sotskov *et al.* (1997) introduce another viewpoint for stability. They handle the uncertainty in a job shop environment by an *a posteriori* analysis, in which an optimal schedule has already been constructed and the challenge is to determine the maximum variation in the processing time of the operations such that the optimal schedule at hand still remains optimal. Such a maximum variation is called the *stability radius* of the schedule. This notion of stability, obtained by sensitivity analysis, can be considered as a measure of *solution robustness* as per Herroelen and Leus (2005). Although this type of post-optimality analysis may provide some valuable insights about the impacts of

uncertainty, it is also associated with some problems. If the stability radius of the optimal schedule is large enough to accommodate all possible changes in the processing times, the optimal schedule at hand can safely be used, but if it is not that large, the question of what course of action to take remains to be answered. Hence, in this paper, we take a proactive stance and incorporate uncertainty into the scheduling processes. We concentrate on optimizing the *quality robustness* rather than the solution robustness.

In this paper, we consider the single-machine scheduling problem with random processing times and machine breakdowns. We define two robustness (*RM*1 and *RM*2) and three stability measures (*SM*1, *SM*2, and *SM*3). In general, calculating actual robustness and stability measures analytically is very difficult. For that reason, in the previous studies researchers employ surrogate measures to indirectly calculate the robustness or stability of a schedule. The surrogate measures used in the existing studies, however, are simple in the sense that they do not adequately incorporate the known information about the uncertainty, as also stated in Mehta and Uzsoy (1998). In this paper, we use the formal probability theory to derive inferences about minimizing robustness or stability measures. Specifically, we solve the problem for a number of special cases. For intractable cases, instead of employing surrogate measures, we use a beam search (BS) algorithm developed in this study that employs simulation to calculate robustness or stability measures. Thus, we use the available information about the uncertainty better than does the indirect approach of employing surrogate measures. Moreover, in the previous studies, makespan or maximum lateness is used as the performance measure for the sake of simplicity. In our study, however, we consider flow time and tardiness criteria, as they are used more often in practice.

3. Proposed probabilistic approach

3.1. *Notation*

We consider the single-machine scheduling problem with random processing times and machine breakdowns. The uptimes have independent and identical general distribution $G_1(t)$. Similarly, the downtimes (i.e., the times that the machine is not in operation due to breakdown) are independent and identically distributed (i.i.d.) according to a general distribution $G_2(t)$. The processing times of the jobs are all random variables with known general distribution functions that may differ from job to job. Let $H_j(t)$ be the processing time distribution of job j. Let C_j denote the completion time of job j in the initial schedule and C_j^{r} denote the completion time of job j in the realized schedule. Let X_j denote the processing time of job j. We assume that all n jobs are released at time $t = 0$. Let $U_1, U_2 \ldots$ be the sequence of uptimes and $D_1, D_2 \ldots$ be the sequence of downtimes. That is, the machine is operational from time 0 until U_1, when the first breakdown occurs. The machine then takes time D_1 to be repaired and is again available for processing from time $U_1 + D_1$ until time $U_1 + D_1 + U_2$, and so on. We denote this stochastic single-machine scheduling problem as $1 \,|X_j \sim H_j(t)$*; brkdwn:* $U \sim G_1(t)$, $D \sim G_2(t)$; $\beta \,|\, \gamma$ where $1 \,|\beta|\gamma$ denotes the deterministic version. Here, β is the set of scheduling attributes, such as release dates, presence of sequence-dependent setup times, preemptions, precedence constraints, etc., and γ is the objective function. If breakdowns were not present, the notation would be $1 \,|X_j \sim H_j(t); \beta|\gamma$.

Define $N(t) = \sup\{k \geq 0 | \sum_{i=0}^{k} U_i \leq t\}$, where $U_0 := 0$. That is, $N(t)$ is the number of machine breakdowns that occur up to total busy time t. Note that $N(t)$ is increasing in t. Here, we consider the case where the machine can be down more than once during the processing of a job and the job is processed for its remaining processing time after each breakdown (i.e., the work done on a job is not lost).

Y_j denotes the time that job j occupies the machine, including the processing time of the job and all the repair times during which the job stays on the machine. Let R_{jk} denote the kth repair time during the processing of job j. Since R_{jk}'s are i.i.d., let $r = E[R_{jk}] = \int_0^\infty \mathrm{td}G_2(t)$ and $v = \mathrm{Var}[R_{jk}] = \int_0^\infty (t-r)^2 dG_2(t)$. Let B_j denote the number of machine failures during the processing of job j. Then, we have $Y_j = X_j + \sum_{k=1}^{B_j} R_{jk}$.

We first begin by a definition and several propositions, which will be used in the treatment of the robustness and stability measures in Sections 3.3 and 3.4, respectively.

3.2. *Preliminaries*

Definition 1 (Ross, 1983)**.** *A random variable V is said to be stochastically larger than a random variable W, written $V \geq_{st} W$, if $P\{V > a\} \{\geq P\} \{W > a\}$ for all a.*

Proposition 1. *Let $V_1, \ldots, V_n$ be independent and $W_1, \ldots, W_n$ be independent. If $V_i \geq_{st} W_i$ for all i, then for any increasing f, $f(V_1, \ldots, V_n) \geq_{st} f(W_1, \ldots, W_n)$.*

Proposition 2. *If $V \geq_{st} W$ then $\max\{V, 0\} \geq_{st} \max\{W, 0\}$.*

We refer the reader to Example 8.2(a) and Question 8.1 in Ross (1983) for the proofs of these two propositions. Both proofs involve the *coupling method*, which is explained in Ross's Chapter 8.

Proposition 3. *If uptimes are exponentially distributed with the rate λ, then*:

$$E[B_j] = \lambda E[X_j],$$
$$E\left[B_j^2\right] = \lambda E[X_j] + \lambda^2 E\left[X_j^2\right],$$
$$\mathrm{Var}[B_j] = \lambda E[X_j] + \lambda^2 \mathrm{Var}[X_j].$$

Proposition 4. *If uptimes are exponentially distributed with the rate* λ, *then*:

$$E[Y_j] = (1+\lambda r)E[X_j],$$
$$\mathrm{Var}[Y_j] = \lambda(v+r^2)E[X_j] + (1+\lambda^2 r^2)\mathrm{Var}[X_j].$$

3.3. *Robustness*

Although there are some studies which measure robustness as a minimax regret (e.g., Daniels and Kouvelis (1995)), the majority of the recent studies on robustness involve expected realized performance. This expected realized performance can be the robustness measure by itself (e.g., Wu *et al.* (1999)) or can be a part of it (e.g., Leon *et al.* (1994)). In this study, we use the expected realized performance measure as the robustness measure. We begin with the flow time case.

3.3.1. *Total flow time*

Recall that $RM1$ is the expected realized total flow time. That is, $RM1 = E[\sum_{j=1}^{n} C_j^{\mathrm{r}}]$. Minimizing expected total weighted flow time in a single-machine environment subject to random machine breakdowns is known to be NP-hard (Adiri *et al.*, 1989). Even though the status of the unweighted case is unknown, it can be said that the problem is analytically intractable, for it is difficult even to calculate the objective function value of a given solution. We present an optimality condition that holds in a special case here.

Theorem 1. *If* $X_j \leq_{st} X_{j+1}$ *for* $j = 1, \ldots, n-1$, *the job sequence* $1, \ldots, n$, *i.e.*, *SSPT* (*Stochastically Smallest Processing Time*) *order, is an optimal solution to* $1 \,|X_j \sim H_j(t)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM1$ *problem.*

Proof. Consider an optimal sequence S. Assume that there exists a pair of adjacent jobs i and j such that $X_j \leq_{st} X_i$ and job j succeeds job i in S. Because if such a pair does not exist, either S is already the sequence $\{1, \ldots, n\}$ or it can be put into that form by simply swapping the labels of the jobs whose processing times have the same distribution. Therefore, without loss of generality we assume that there exists such a pair. Now consider a sequence S', obtained from S by swapping the positions of jobs i and j. We compare $RM1(S)$ and $RM1(S')$. We may ignore the jobs other than i and j in this comparison, since nothing changes for them. Let their contribution to the objective function be c. Let T denote the sum of the processing times of the jobs that precede i in S. We have:

$$RM1(S) = E\left[T + X_i + \sum_{k=1}^{N(T+X_i)} D_k + T + X_i + X_j + \sum_{k=1}^{N(T+X_i+X_j)} D_k\right] + c,$$

and

$$RM1(S') = E\left[T + X_j + \sum_{k=1}^{N(T+X_j)} D_k + T + X_i + X_j + \sum_{k=1}^{N(T+X_i+X_j)} D_k\right] + c.$$

Hence,

$$RM1(S) - RM1(S') = E[X_i - X_j] + E\left[\sum_{k=1}^{N(T+X_i)} D_k - \sum_{k=1}^{N(T+X_j)} D_k\right].$$

Since $X_j \leq_{st} X_i$ and $N(t)$ is increasing, $N(T+X_j) \leq_{st} N(T+X_i)$ by Proposition 1. By coupling we also have $\sum_{k=1}^{N(T+X_j)} D_k \leq_{st} \sum_{k=1}^{N(T+X_i)} D_k$, and therefore $RM1(S) - RM1(S') \geq 0$. This means that $S\prime$ is also an optimal solution. If we continue interchanging positions of adjacent jobs in this manner until no pair of adjacent jobs i and j such that $X_j \leq_{st} X_i$ and job j succeeds job i exists, we obtain a series of optimal solutions. The last solution we obtain is either already the sequence $\{1, \ldots, n\}$ or it can be put into that form by simply swapping the labels of the jobs whose processing times have the same distribution. ■

Corollary 1. *SEPT order gives an optimal solution for* $1 \mid X_j \sim$ *exponential*(λ_j); *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM1$.

Corollary 2. *SEPT order gives an optimal solution for* $1 \mid X_j \sim H_j(t) \mid RM1$.

Corollaries 1 and 2 are known results in the literature. See Pinedo (2002, Chapter 10). Theorem 1 can also be deducted from the dominance rule developed by Li and Glazebrook (1998).

3.3.2. *Total tardiness*

$RM2$ is the expected realized total tardiness. That is, $RM2 = E[\sum_{j=1}^{n} \max(0, C_j^{\mathrm{r}} - d_j)]$, where d_j is the due date of job j.

Theorem 2. $1 \mid X_j \sim H_j(t)$; *brkdwn:* $U \sim G_1(t)$, $D \sim G_2(t) \mid RM2$ *is NP-hard.*

Proof. We reduce $1||\sum_j T_j$ to $1 \mid X_j \sim H_j(t)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM2$. Begin with a $1||\sum_j T_j$ instance. Take all repair times as zero. Do not change processing times, i.e, $H_j(t)$ and $G_2(t)$ are degenerate distributions. Take $G_1(t)$ as any arbitrary distribution. Due dates also do not change. An optimal solution to this newly constructed $1|X_j \sim H_j(t)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM2$ instance is also an optimal solution to the original $1||\sum_j T_j$ instance. $1||\sum_j T_j$ is known to be NP-hard (Du and Leung, 1990) and the result follows. ■

Theorem 3 (Dominance rule). *Consider the* $1|X_j \sim H_j(t)$ $|$ *RM*2 *problem. For any two jobs i and j if* $X_i \leq_{st} X_j$ *and* $d_i \leq d_j$, *then there exists an optimal sequence in which job i precedes job j.*

Proof. Can be proven with an interchange argument similar to the proof of Theorem 1. Detailed proof makes use of Propositions 1 and 2 for the jobs in between and compares the expected tardiness values of jobs i and j (Goren and Sabuncuoglu, 2009). ■

Corollary 3. *Consider the* $1|X_j \sim$ *exponential*(λ_j) $|$ *RM2 problem. If due dates are agreeable, i.e., if the Earliest Due Date* (*EDD*) *first and SEPT sequences are the same, the EDD sequence is optimal.*

Corollary 4. *SEPT order gives an optimal solution for* $1 \mid X_j \sim$ *exponential* (λ_j); $d_j = d \mid RM2$.

Note that if the processing times are exponentially distributed, Theorem 3 can be extended to include arbitrary machine breakdowns. As a result, Corollaries 3 and 4 are also still valid in the presence of machine breakdowns. For the proofs of the last two corollaries and the inclusion of the breakdown process, we refer the reader to Pinedo (2002), Section 10.4.

3.4. *Stability*

Recall that a stable schedule is one that should not deviate much from the initial schedule. The deviation is generally measured in terms of the differences between the job completion times in the initial and realized schedules. Hence, a typical stability measure is a non-decreasing function of the deviation of job completion times. We use three stability measures called *SM*1, *SM*2, and *SM*3. *SM*3 was already available in the literature. *SM*1 and *SM*2 are proposed for the first time in our study.

3.4.1. *Stability measure 1 (SM1) and stability measure 2 (SM2)*

Recall that *SM*1 is the expected sum of squares of job completion time differences between the initial and realized schedules. That is, $SM1 = E[\sum_{i=1}^{n} (C_i - C_i^r)^2]$. A scheduler who is aware of the fact that initial schedules will inevitably deviate due to random disruptions can prepare his/her secondary plans according to expected completion times rather than deterministic completion times. In this case, a reasonable stability measure can be

$$SM2 = E\left[\sum_{i=1}^{n} \left(E[C_i^r] - C_i^r\right)^2\right] = \sum_{i=1}^{n} \mathrm{Var}[C_i^r].$$

Theorem 4 (SVPT (Smallest Variance of Processing Time first) Optimality). *If* $\mathrm{Var}[X_j] \leq \mathrm{Var}[X_{j+1}]$ *for* $j = 1, \ldots, n-1$, *the job sequence* $\{1, \ldots, n\}$ *is an optimal solution to the* $1 \mid X_j \sim H_j(t) \mid SM1(SM2)$ *problem. In other words, the SVPT rule gives an optimal solution.*

Proof. The proof is by contradiction. Let S be an optimal sequence but assume that there exists a pair of adjacent jobs i and j such that $\mathrm{Var}[X_i] > \mathrm{Var}[X_j]$ and job j succeeds job i in S. Now consider a sequence S', obtained from S by swapping the positions of jobs i and j. We compare $SM1(S)$ and $SM1(S')$. We may ignore the jobs other than i and j in this comparison, since nothing changes for them. Let their contribution to the objective function be c. Let T denote the sum of the processing times of the jobs that precede i in S. We have:

$$\begin{aligned} SM1(S) &= E\left[(T + X_i - E[T + X_i])^2\right] + E[(T + X_i + X_j \\ &\quad - E\left[T + X_i + X_j\right])^2] + c \\ &= \mathrm{Var}[T + X_i] + \mathrm{Var}[T + X_i + X_j] + c \end{aligned}$$

and

$$\begin{aligned} SM1(S') &= E\left[(T + X_j - E[T + X_j])^2\right] + E[(T + X_i + X_j \\ &\quad - E[T + X_i + X_j])^2] + c \\ &= \mathrm{Var}[T + X_j] + \mathrm{Var}[T + X_i + X_j] + c. \end{aligned}$$

Hence,

$$SM1(S) - SM1(S') = \mathrm{Var}[X_i] - \mathrm{Var}[X_j].$$

Since $\mathrm{Var}[X_i] > \mathrm{Var}[X_j]$, $SM1(S) > SM1(S')$. That is, there is a strict improvement in the objective function after the interchange. This contradicts the fact that S is an optimal solution. The proof for the *SM*2 performance measure can be done similarly. ■

Corollary 5. *SEPT solves* $1 \mid X_j \sim$ *exponential* $(\lambda_j) \mid SM1(SM2)$ *optimally.*

Theorem 5 (SEPT Optimality). *If* $E[X_i] > E[X_j]$ *implies* $Var[X_i] \geq Var[X_j], \forall (i, j)$ *then* $1 \mid X_j \sim H_j(t)$; *brkdwn*: $U \sim$ *exponential*(λ), $D \sim G_2(t) \mid SM1(SM2)$ *is solved optimally by the SEPT rule.*

Proof. See Appendix. ■

Corollary 6. *SEPT solves*:
$1 \mid X_j \sim$ *exponential*(λ_j); *brkdwn*: $U \sim$ *exponential*(λ), $D \sim G_2(t) \mid SM1(SM2)$ *optimally.*

$1 \mid X_j \sim H_j(t)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid SM1(SM2)$ *is analytically intractable in the general case.*

3.4.2. *Stability measure 3 (SM3)*

*SM*3 is the expected absolute job completion time differences between the initial and realized schedules:

$$SM3 = E\left[\sum_{i=1}^{n} \left|C_i - C_i^r\right|\right].$$

This kind of measuring of the deviation between two schedules was first proposed by Wu *et al.* (1993).

Theorem 6 (SPT Optimality). *SPT solves*:

Table 1. Analytically tractable cases; robustness

Problem	*Algorithm*	*Theorem/ corollary*
$1 \mid X_j \sim H_j(t)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM1$	SSPT	Theorem 1
$1 \mid X_j \sim exponential(\lambda_j)$; *brkdwn*: $U \sim G_1(t)$, $D \sim G_2(t) \mid RM1$	SEPT	Corollary 1
$1 \mid X_j \sim H_j(t) \mid RM1$	SEPT	Corollary 2
$1 \mid X_j \sim H_j(t) \mid RM2$	Dominance rule	Theorem 3
$1 \mid X_j \sim exponential(\lambda_j) \mid RM2$	EDD if EDD and SEPT sequences are the same	Corollary 3
$1 \mid X_j \sim exponential(\lambda_j)$; $d_j = d \mid RM2$	SEPT	Corollary 4

$1 \mid p_j$, *brkdwn*: $U \sim exponential(\lambda)$, $D \sim G_2(t) \mid SM3$ *optimally where* p_j *denotes the deterministic processing time of job j.*

Proof. Can easily be proven with an interchange argument.

$1 \mid X_j \sim H_j(t)$; *brkdwn:* $U \sim G_1(t)$, $D \sim G_2(t) \mid SM3$ is analytically intractable in the general case. ■

The results are summarized in Tables 1 and 2.

4. A branch-and-bound algorithm for $1 \mid X_j \sim H_j(t) \mid RM2$

In this section, we first focus on the $1 \mid X_j \sim H_j(t) \mid RM2$ problem because of two reasons: first, the total tardiness performance measure is popular and frequently used in practice. Second, we have a dominance rule (Theorem 3) that can be effectively used in a branch-and-bound (B&B) algorithm to keep the size of the search tree manageable.

Table 2. Analytically tractable cases; stability

Problem	*Algorithm*	*Theorem/ corollary*
$1 \mid X_j \sim H_j(t) \mid SM1(SM2)$	SVPT	Theorem 4
$1 \mid X_j \sim exponential(\lambda_j) \mid SM1(SM2)$	SEPT	Corollary 5
$1 \mid X_j \sim H_j(t)$; *brkdwn*: $U \sim exponential(\lambda)$, $D \sim G_2(t) \mid SM1(SM2)$	SVPT if $E[X_i] > E[X_j]$ implies $\mathrm{Var}[X_i] \geq \mathrm{Var}[X_j], \forall(i, j)$	Theorem 5
$1 \mid X_j \sim exponential(\lambda_j)$; *brkdwn*: $U \sim exponential(\lambda)$, $D \sim G_2(t) \mid SM1(SM2)$	SEPT	Corollary 6
$1 \mid p_j$, *brkdwn*: $U \sim exponential(\lambda)$, $D \sim G_2(t) \mid SM3$	SPT	Theorem 6

The algorithm developed in this section is for the problems where the processing time distributions of any two jobs are stochastically comparable. Typical examples are normal distribution with a common coefficient of variation (c_v), gamma distribution with the same scale parameter, and Poisson distribution. For all these distributions, ordering in the expected value corresponds to ordering in the stochastic sense. Moreover, the job completion times in any sequence also have the same type of distributions as the processing time distributions; i.e., they belong to the same family.

We should be very careful when processing time distributions are normal with a common coefficient of variation because stochastic comparability is only valid for the non-negative part of the distributions. Thus, the probability of having negative processing times should be negligibly small ($c_v < 1/3$) for a satisfactory performance of the algorithm.

In the proposed B&B algorithm, we develop the schedules progressively in the forward direction. At level k of the B&B tree, jobs in the first k positions are specified. We use the dominance rule in Theorem 3 during the branching process. The initial upper bound is taken as the minimum expected total tardiness value of the SPT, EDD, and ATC solutions. The upper bound of each node is taken as the expected total tardiness of the EDD completion of that node. If the global upper bound is greater than the upper bound of a node, it is updated. There are two lower bounds considered: a loose one and a tight one. These are explained in the next two theorems. If the lower bound of a node is greater than the global upper bound, it is pruned. We use a most-promising-node-first exploration strategy; that is, the node with the lowest upper bound value is branched first.

Theorem 7 (Lower Bound 1)**.** *Consider the problem* $1 \mid X_j \sim H_j(t) \mid RM2$*. Arrange the due dates in non-decreasing order and assign them to the jobs arranged in a stochastic non-decreasing order of processing times* (*assuming all jobs can be ordered stochastically*)*. The optimal expected total tardiness value of this new problem* $P1$ *is a lower bound on the optimal objective value of the original problem* $1 \mid X_j \sim H_j(t) \mid RM2$*.*

Proof. The proof is by an interchange argument. The optimal solution to $P1$ is an EDD sequence by Theorem 3. We now show that its objective function value is a lower bound on the optimal objective function value of the original problem. We begin by an optimal solution S^* of the original problem and convert it to an optimal solution of $P1$. The procedure is as follows:

Step 1. Consider every adjacent job pair. If a job with a greater due date precedes a job with a smaller due date, swap their due dates but not their positions; just assign the due date of the former job to the latter job and *vice versa*. Continue in this fashion until all due dates are in non-decreasing order. Each swap of the due dates results in a possible decrease

in the expected total tardiness of the schedule but never an increase.

Step 2. Take the resultant schedule of Step 1 as the input and process it in the same way as in Step 1. The only difference is, instead of due dates, compare and exchange the processing times of the adjacent job pairs if necessary. ■

The resulting schedule is optimal for $P1$ and its objective function value is a lower bound on that of the original problem. The deterministic version of this lower bound is developed by Chu (1992).

Della Croce *et al.* (1998) propose another lower bound for the deterministic problem. In this study, we extend this lower bound to the stochastic problem as follows:

Theorem 8 (Lower Bound 2). *Consider the* $1 \mid X_j \sim H_j(t) \mid RM2$ *problem. Relabel the jobs according to the non-decreasing stochastic order of their processing times (assuming all jobs can be ordered stochastically). That is, the job with the SSPT is job* 1, *and with the stochastically largest processing time is job n. Split the job set J into two subsets* $J_1 = 1, \ldots, l$ *and* $J_2 = l + 1, \ldots, n$, *where* $l = \lfloor n/2 \rfloor$. *For each subset, separately arrange the due dates in non-decreasing order and assign them to the jobs arranged in a stochastic non-decreasing order of processing times. The optimal expected total tardiness value of this new problem P2 is a lower bound on that of the original problem* $1 \mid X_j \sim H_j(t) \mid RM2$.

The proof of the theorem basically involves the same interchange argument as in the proof of Theorem 7. The only difference is that we apply Steps 1 and 2 separately to the jobs in subsets J_1 and J_2. That is, at each pass of Step 1 or Step 2, we examine successive jobs (not necessarily adjacent) that belong to the same subset. At the end, we obtain a *feasible* schedule to $P2$, whose expected total tardiness is no greater than the optimal objective value of the original problem. The expected total tardiness value of an *optimal* solution to $P2$ is possibly even less, so it is a lower bound for the original problem. The optimal solution to $P2$ can be found in polynomial time. For two solution procedures, each with $O(n^2)$ time complexity, the reader can refer to Della Croce *et al.* (1998).

5. A BS algorithm for other intractable problems

The proposed B&B algorithm relies on Theorem 3 as a dominance rule and Theorems 9 and 10 as lower bounds. These theorems are valid under the assumption that for any two jobs, their processing times are stochastically comparable. Also, machine breakdowns are not considered. In this section, we develop a BS algorithm that can be used with any processing time distribution and any objective function ($RM1$, $RM2$, $SM1$, $SM2$, or $SM3$) and that can also handle a general machine breakdown/repair process.

BS is an approximate B&B method which operates on a search tree. BS has been used to solve combinatorial optimization problems for the last two decades. There are several successful applications to job shop scheduling and flexible manufacturing systems scheduling problems with static and deterministic assumptions (Sabuncuoglu and Karabuk, 1998; Sabuncuoglu and Bayiz, 1999). Generally speaking, BS is similar to a breadth-first search as since it progresses level by level without backtracking. However, unlike breadth first, only the best β (*beam width*) promising nodes are kept for further branching at any level. The potential promise of each node is determined by a *global evaluation function*, which typically estimates the minimum total cost of the best solution obtained from the partial schedule represented by the node.

In a BS implementation, the beams may progress independently (i.e., at all levels other than level 1, each of β promising nodes has a different ancestor), but in our implementation, we use dependent beams (i.e., at each level, all the descendants are evaluated and the best β of them are chosen without paying attention to their ancestors). Specifically, we first complete the partial schedule that the node represents according to the objective function in use. If the objective function is $RM1$, the schedule is completed according to the SEPT rule. Similarly, ATC is used for $RM2$, and SVPT is used for $SM1$, $SM2$, or $SM3$. We then simulate the resulting schedule ten times. The average of these objective function values is taken as the global evaluation function value. The simulations are done with the help of a simple discrete-event simulation model coded in C++ language. First, the processing times of the jobs are generated according to their respective probability distributions. After that, the machine uptimes and downtimes are generated and inserted into their proper positions in the schedule. Finally, the realized job completion times are obtained and used for the performance measure calculations.

6. Computational experiments

The performance of the proposed algorithms was measured on a non-dedicated Linux box with dual AMD Opteron 2.6 GHz CPUs and 2 GBs of physical memory. The codes were written in C++ language. The data generation scheme, initially proposed by Fisher (1976), is explained in the next section.

6.1. *Test problems and BS parameters*

The problem instances of varying degrees of difficulty are generated by means of two factors: tardiness factor (TF) and range of due dates (DR). For each problem, first the processing time means are generated from a uniform distribution with parameters (1, 100). Then the due dates are

Table 3. Experimental environment

Processing time mean ($E[X_j]$)	$U[1, 100]$
Number of jobs (n)	10, 20, 30, 40, 50, 75, 100
Due dates (d_j)	$U[P(1 - TF - DR/2), P(1 - TF + DR/2]$, where P is the sum of processing time means TF in {0.2, 0.4, 0.6, 0.8} DR in {0.2, 0.4, 0.6, 0.8, 1.0}

generated from a uniform distribution, which depends on the sum of the processing time mean (P) and on R and T. The due date distribution is uniform over [P(1 – TF – DR/2), P(1 – TF + DR/2)]. The values of TF and DR are selected from {0.2, 0.4, 0.6, 0.8} and {0.2, 0.4, 0.6, 0.8, 1.0}, respectively. This yields 20 combinations of TF, DR for each problem size. The number of jobs n is selected from the set {10, 20, 30, 40, 50, 75, 100}. Ten different instances were solved for each setting of n, TF, DR, which gives 200 instances for each choice of n. For the B&B algorithm, problems up to the size of ten ($n = 10$) were solved. For the BS algorithm, all problem sizes were solved for each objective function ($RM1$, $RM2$, $SM1$, $SM2$, and $SM3$). Table 3 summarizes the experimental settings.

We call each combination of n, TF, and DR a *problem class*. We also assign each problem class a code name: *probxyz*. Here x, y, z are the levels of the n, TF, and DR factors, respectively. For example, prob231 is the problem class in which $n = 20$, $TF = 0.6$, and $DR = 0.2$.

The beam width was taken as four. Recall that the proposed BS algorithm employs simulation as the global evaluation function. During simulation runs, we used a gamma distribution as a busy-time distribution with a shape parameter of 0.7 and a scale parameter to be specified. We used a gamma distribution with a shape parameter of 1.4 for the downtime distribution, as recommended by Law and Kelton (2000). The scale parameter of the busy-time distribution was arranged so that the mean was 300. Similarly, the scale parameter of the downtime distribution was arranged so that the mean was 50.

6.2. *Evaluation of the algorithms for $1 \mid X_j \sim H_j(t) \mid RM2$*

Recall that the B&B algorithm was developed for the $1 \mid X_j \sim H_j(t) \mid RM2$ problem (i.e., there are no machine breakdowns) and the processing time distributions of any two jobs can be stochastically compared. We took the processing time distribution of each job as a gamma distribution, with a scale parameter of two. The shape parameters were arranged such that the mean processing times equaled the previously generated values (see Section 6.1). Only 200 ten-job problems were solved because of the computational time limitations. Each problem instance was solved two times, once using Lower Bound 1 (loose) and once using Lower Bound 2 (tight). Table 4 presents the results. In Table 4, a better CPU time is marked with an asterix (*) for each problem class.

We observe two things: (i) Generally, as TF and DR increase, the problems get easier and require less computational time to solve (i.e., the problems with loose due dates are harder to solve); and (ii) the extra computational time required for calculating a tight lower bound pays off for hard problem classes, but this is not worth the effort for easy problem classes.

Tables 5 and 6 present the average number of pruned nodes due to the dominance rule for loose and tight lower bounds, respectively. We can observe that the dominance rule works quite effectively.

For example, on average, 7.1 nodes are pruned due to the dominance rule among the ten nodes in level 1. Among $(10 - 7.1) \times 9 = 26.1$ nodes in level 2, the dominance rule prunes 15.1 and 13.7 for the algorithms with loose and tight lower bounds, respectively.

We also observe that the dominance rule prunes fewer nodes in each level for the algorithm with the tight lower bound, but this is expected because for this case, more nodes are pruned due to their upper and lower bound comparisons.

Table 4. Results obtained using the B&B algorithm

	CPU time				*CPU time*		
Problem	*Loose*	*Tight*	*Objective*	*Problem*	*Loose*	*Tight*	*Objective*
prob111	1962.23	1833.26*	75.64	prob131	292.81*	432.88	487.65
prob112	778.20*	1255.01	36.72	prob132	205.67*	276.43	593.44
prob113	318.40*	507.28	27.16	prob133	133.51*	171.97	596.93
prob114	557.84*	671.69	34.41	prob134	346.47*	417.92	487.82
prob115	479.59*	670.11	23.18	prob135	646.03	575.50*	707.64
prob121	849.89	816.65*	305.25	prob141	65.91*	144.03	1129.50
prob122	800.41*	1055.06	210.04	prob142	68.32*	148.94	1351.72
prob123	956.70	695.33*	199.16	prob143	91.42*	192.63	1410.66
prob124	323.33*	472.76	124.38	prob144	141.06*	196.17	1140.40
prob125	645.20	593.21*	124.64	prob145	78.82*	93.82	1297.68

Table 5. Performance of dominance rule, loose lower bound

	Nodes pruned—loose								
Problem	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Level 5*	*Level 6*	*Level 7*	*Level 8*	*Level 9*
prob111	7.5	11.3	29.9	78.8	196.5	462.2	800.4	758.9	0.1
prob112	7.1	8.7	25.1	67.0	160.2	296.5	317.4	195.6	0.0
prob113	7.6	12.1	21.7	44.9	89.2	135.8	141.9	70.2	0.2
prob114	7.0	12.5	29.0	69.8	157.8	200.1	183.6	65.9	0.0
prob115	6.3	13.9	38.0	93.0	183.4	242.0	147.9	38.3	0.0
prob121	7.2	17.3	38.9	95.4	202.5	330.3	386.7	167.7	0.1
prob122	6.9	18.2	47.0	109.5	226.1	414.9	380.5	196.8	0.1
prob123	7.1	17.3	43.7	103.8	231.4	420.0	447.8	154.8	0.1
prob124	7.3	16.3	36.2	66.3	115.9	150.0	103.0	51.0	0.0
prob125	6.8	16.7	40.8	95.4	201.6	309.0	233.8	59.4	0.0
prob131	7.0	16.7	40.8	95.5	176.8	199.6	59.7	4.2	0.0
prob132	7.6	15.8	29.6	59.8	93.6	100.0	50.9	11.4	0.0
prob133	8.0	13.7	21.5	40.9	61.1	45.8	15.9	2.5	0.0
prob134	7.0	16.9	38.4	92.1	182.5	222.3	115.5	30.7	0.0
prob135	6.6	19.4	46.7	98.6	156.2	161.4	137.3	27.3	0.3
prob141	6.6	18.7	37.6	50.3	46.5	19.4	8.1	5.4	0.7
prob142	7.1	14.5	26.0	28.4	21.6	10.5	5.9	3.5	0.0
prob143	7.5	11.5	17.4	26.7	29.1	19.0	6.5	3.9	0.0
prob144	6.7	17.9	35.1	59.2	61.2	41.1	21.7	5.5	0.5
prob145	7.5	13.3	20.6	32.4	41.5	43.5	22.8	3.3	0.0
Average	7.1	15.1	33.2	70.8	131.7	191.2	179.4	92.8	0.1

The same 200 problems were also solved by the proposed BS algorithm for comparison. The solutions obtained from the BS were evaluated by the exact objective function, which is also used in the B&B algorithm (i.e., the reported results are not simulation values). Table 7 summarizes the results. Optimal objective function values and minimum CPU times obtained from the B&B algorithm are also included in Table 7. The results indicate that the BS algorithm finds the optimal solution for 51 of the 200 problem instances. The deviation from

Table 6. Performance of dominance rule, tight lower bound

	Nodes pruned—tight								
Problem	*Level 1*	*Level 2*	*Level 3*	*Level 4*	*Level 5*	*Level 6*	*Level 7*	*Level 8*	*Level 9*
prob111	7.5	10.0	17.9	49.3	131.9	310.0	340.6	192.6	0.0
prob112	7.1	8.7	20.6	52.1	126.6	234.5	277.9	126.2	0.0
prob113	7.6	10.9	20.5	38.9	63.2	110.3	87.4	12.8	0.2
prob114	7.0	12.5	20.4	43.0	87.3	126.4	125.4	19.8	0.0
prob115	6.3	12.0	32.5	64.8	106.6	73.4	43.1	0.0	0.0
prob121	7.2	16.0	33.6	74.0	133.2	135.9	76.9	16.0	0.1
prob122	6.9	17.2	41.4	84.9	159.0	179.2	140.5	47.9	0.1
prob123	7.1	13.9	32.4	63.5	102.6	79.0	57.8	3.0	0.1
prob124	7.3	13.0	24.3	41.1	56.3	75.9	32.5	19.8	0.0
prob125	6.8	13.2	30.9	61.6	91.3	92.7	41.6	11.6	0.0
prob131	7.0	15.5	36.6	78.6	120.5	62.9	20.0	3.3	0.0
prob132	7.6	15.8	28.4	44.5	61.5	38.7	14.7	5.2	0.0
prob133	8.0	13.0	17.2	24.8	26.5	16.4	8.7	2.4	0.0
prob134	7.0	16.9	30.0	55.8	80.8	54.9	34.5	7.9	0.0
prob135	6.6	17.2	32.0	48.5	61.0	54.9	21.1	3.4	0.3
prob141	6.6	18.0	34.7	42.0	36.7	17.7	8.1	5.4	0.7
prob142	7.1	13.1	21.3	19.4	15.9	7.9	5.9	3.5	0.0
prob143	7.5	10.0	15.8	21.3	22.9	14.4	5.1	3.9	0.0
prob144	6.7	16.6	27.6	38.3	27.4	13.6	12.4	5.3	0.5
prob145	7.5	11.2	15.9	18.2	20.3	12.9	10.3	1.6	0.0
Average	7.1	13.7	26.7	48.2	76.6	85.6	68.2	24.6	0.1

Table 7. A comparison of the B&B and BS algorithm

Problem	BS CPU time	B&B CPU time	BS objective	Optimal objective	Deviation from the optimal (%)
prob111	2.74	1833.26	77.50	75.64	2.46
prob112	2.93	778.20	37.18	36.72	1.26
prob113	2.55	318.40	30.42	27.16	12.01
prob114	2.65	557.84	34.88	34.42	1.34
prob115	2.71	479.59	23.47	23.18	1.26
prob121	2.86	816.65	307.22	305.25	0.65
prob122	2.68	800.41	212.92	210.04	1.37
prob123	2.80	695.33	207.62	199.16	4.25
prob124	2.80	323.33	128.84	124.38	3.58
prob125	2.61	593.21	125.41	124.64	0.62
prob131	2.22	292.81	489.71	487.65	0.42
prob132	2.93	205.67	599.64	593.44	1.04
prob133	3.20	133.51	601.64	596.93	0.79
prob134	2.63	346.47	492.31	487.82	0.92
prob135	2.71	575.50	709.03	707.64	0.20
prob141	2.57	65.91	1130.89	1129.50	0.12
prob142	2.99	68.32	1355.79	1351.72	0.30
prob143	3.01	91.42	1415.11	1410.66	0.32
prob144	2.78	141.06	1142.29	1140.40	0.17
prob145	2.94	78.82	1297.71	1297.68	0.00

the optimal values is under 2% for most of the problem classes. A paired t-test with $\alpha = 0.05$ indicated that the differences in objective function are statistically significant for only the problem classes prob122 and prob132.

We can conclude that the proposed BS algorithm performs quite satisfactorily for the $1 \mid X_j \sim H_j(t) \mid RM2$ problem and, if computational time is an issue, it can be safely used to generate schedules instead of the exact algorithm.

We also compared the performances of the BS algorithm and the ATC dispatching rule. Table 8 presents a summary of the results. The objective function values reported in this table are the averages of the simulated total tardiness values of the schedules generated by the algorithms. We observe that the BS algorithm performs better and all the differences are found to be significant by a paired t-test with $\alpha = 0.05$.

Table 8. Beam search vs. ATC for $RM2$ no breakdown; summary

	BS			
Number of jobs	CPU time	Objective	ATC objective	Deviation (%)
10	1.22	527.01	547.48	3.88
20	10.31	1760.17	1804.69	2.53
30	34.47	3579.49	3652.44	2.04
40	83.63	6145.22	6253.40	1.76
50	164.13	9393.70	9562.67	1.80
75	556.58	20 723.12	21 088.92	1.77
100	1323.28	35 926.39	36 466.31	1.50

6.3. *Evaluation of proposed BS algorithm for other intractable problems with machine breakdowns*

The performance of the proposed BS algorithm was evaluated by solving numerous problem instances for each objective function. Since $RM1$, $SM1$, $SM2$, and $SM3$ are not due-date-related performance measures, ten instances from probx11 ($x = 1, \ldots, 7$) classes were used during the experiments, giving rise to 70 problem instances for each objective function. In other words, tardiness factor (TF) and range of due dates (DR) do not vary among test problems because they are irrelevant. For $RM2$, ten instances from probxyz ($x = 1, \ldots, 7$, $y = 1, \ldots, 4$, $z = 1, \ldots, 5$) classes were used, yielding a total of 1400 problem instances. All problems include machine breakdown/repair. Since these problems are analytically intractable we do not know their optimal solutions. Thus, we compared the performance of the proposed BS algorithm to a priority dispatching rule for each objective function. The dispatching rule that was used depended on the objective function. For example, if the objective function is $RM1$, SEPT is used. Similarly, SVPT is used for the stability measures ($SM1$, $SM2$, or $SM3$). Note that SEPT is optimal for $RM1$ and SVPT is optimal for $SM1$, $SM2$, or $SM3$ under special conditions (see Theorem 1 and Theorems 4 to 8). However, we expect these dispatching rules to also perform well under more general conditions (even if the stated optimality conditions in Theorems 1 and 4 to 8 do not hold).

For $RM2$, three dispatching rules and three versions of the BS algorithm were considered. The first dispatching rule was ATC: at every time point t the machine becomes free, a priority index is calculated for each unscheduled job j, and the job with the highest priority is scheduled next. Note that the priority indices were calculated only at the deterministic completion times of the jobs. Additionally, two proactive versions of ATC, namely ProATC1 and ProATC2, were developed to incorporate the machine breakdown and repair information. In ProATC1, a job's processing time is inflated by the expected repair duration during the processing of that job. Specifically, the processing time for job j was taken as

$$p_j = E[X_j] + \frac{E[X_j]}{E[U]} E[D] = E[X_j]\left(1 + \frac{E[D]}{E[U]}\right).$$

The priorities of the jobs were calculated using these new processing time values. In ProATC2, the time points where the priority indices were calculated were adjusted to include machine breakdowns. We anticipate a constant downtime period ($E[D]$) after every constant busy-time period ($E[U]$). That is, time (t) is advanced by $E[D]$ every time the machine stays up for $E[U]$. The BS algorithms under consideration were the *classical BS*, *simulation-based BS*, and *proactive BS*. In classical BS, the global evaluation function is the regular total tardiness measure. At each level of the search

Table 9. Comparison of algorithms, non-due-date-related, gamma repair times

Objective function	*Number of jobs*	*BS*		*Surrogate (BS-M1)*		*Dispatching rule (SEPT/SVPT)*	
		CPU time	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*
*RM*1	10	0.09	2092.52	0.00	2091.84*	0.00	2103.06^{+}
	20	0.77	9116.38*	0.02	9123.58	0.00	9154.53^{+}
	30	2.52	18 244.30	0.08	18 237.20*	0.00	18 430.70^{+}
	40	6.11	32 474.30*	0.20	32 487.80	0.00	33 240.40^{+}
	50	11.65	48 973.80*	0.36	48 992.00	0.00	50 259.40^{+}
	75	39.23	110 345.00*	1.36	110 373.00	0.00	114 193.00^{+}
	100	93.35	191 219.00	3.13	190 989.00*	0.00	199 043.00^{+}
*SM*1	10	0.07	126 503.00*	0.00	276 603.00^{+}	0.00	134 952.00
	20	0.57	643 599.00*	0.02	966 289.00^{+}	0.00	697 516.00
	30	1.83	1 265 710.00*	0.07	1 411 240.00^{+}	0.00	1 360 740.00
	40	4.41	2 505 680.00*	0.19	2 961 280.00	0.00	3 158 530.00^{+}
	50	8.45	3 413 200.00*	0.34	4 090 790.00	0.00	4 684 200.00^{+}
	75	28.43	8 049 690.00*	1.24	8 890 310.00	0.00	10 053 000.00^{+}
	100	67.30	15 844 300.00*	2.80	16 313 500.00	0.00	22 206 400.00^{+}
*SM*2	10	0.07	122 703.00*	0.00	270 075.00^{+}	0.00	131 267.00
	20	0.58	608 366.00*	0.02	915 720.00^{+}	0.00	648 840.00
	30	1.84	1 099 120.00*	0.08	1 224 390.00	0.00	1 249 160.00^{+}
	40	4.43	2 180 560.00*	0.18	2 545 270.00	0.00	2 668 970.00^{+}
	50	8.50	2 860 190.00*	0.34	3 382 500.00	0.00	3 692 810.00^{+}
	75	28.60	6 112 700.00*	1.26	6 776 440.00	0.00	7 311 420.00^{+}
	100	67.71	11 019 900.00*	2.81	11 458 700.00	0.00	15 095 700.00^{+}
*SM*3	10	0.07	678.40*	0.00	1101.34^{+}	0.00	716.07
	20	0.58	2280.15*	0.02	3049.16^{+}	0.00	2400.81
	30	1.83	3904.92*	0.08	4372.56^{+}	0.00	4216.34
	40	4.42	6150.28*	0.19	7224.14^{+}	0.00	7043.67
	50	8.49	8226.24*	0.34	9 508.64^{+}	0.00	9757.92
	75	28.43	15 374.30*	1.25	16 827.40	0.00	17 881.30^{+}
	100	67.29	25 692.50*	2.88	26 264.00	0.00	31 096.80^{+}

tree, partial schedules in the nodes are completed by the ATC rule, and β nodes with the smallest total tardiness values are retained while the others are pruned permanently. Note that classical BS does not consider breakdowns or processing time variability. Simulation-based BS is like classical BS, except that it employs simulation as the global evaluation function; therefore, processing time variability and machine breakdowns are considered. In proactive BS, similar to simulation-based BS, the global evaluation function is based on simulation. The only difference is that in simulation-based BS, the partial schedules in the nodes are completed by the ATC rule before global evaluation, whereas in proactive BS they are completed by ProATC2.

To observe the effect of using simulation instead of surrogate measures, the same problem instances were also solved with a variant of the proposed BS algorithm (BS-M1) for each objective function. The most frequently used surrogate measure in the literature is the average slack method developed by Leon *et al.* (1994). This measure was developed for a job shop environment with the makespan measure. The measure depends on job *slacks*, which is defined as the amount of time that a job's processing can be delayed without increasing the makespan of the schedule. Since in this study we operate in a single-machine environment with all jobs present at time $t = 0$, slacks for all jobs are zero and a slack-based measure cannot be applied. There are other surrogate measures that require inserting idle times, as in Mehta and Uzsoy (1998). Since our solution space is the class of non-delay schedules, these types of surrogate measures are not quite applicable. BS-M1 uses the Method 1 surrogate measure in Goren and Sabuncuoglu (2008) to globally evaluate the nodes instead of simulation. Method 1 assumes that the machine fails after every constant busy-time period of length $\lambda L + (1 - \lambda)U$, where λ is a real number between zero and one, and L and U are the 25th and 975th 1000-tiles of the busy-time distribution $G_1(t)$. It was also assumed that all repair activities last for a time period of length r, the expectation of the repair time distribution $G_2(t)$. Method 1 was developed for an environment where the job processing times are deterministic. To use it as a global evaluator, we further assumed that the job processing times are deterministic and their values are equal to the expectations of the respective processing

Table 10. Comparison of algorithms, non-due-date-related, exponential repair times

Objective function	*Numbers of jobs*	*BS*		*Surrogate (BS-M1)*		*Dispatching rule (SEPT/SVPT)*	
		CPU time	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*
*RM*1	10	0.08	2137.29	0.00	2135.48*	0.00	2154.75^{+}
	20	0.63	9336.27	0.02	9327.52*	0.00	9483.38^{+}
	30	2.07	18 852.60	0.08	18 839.60*	0.00	19 219.10^{+}
	40	4.99	33 646.20	0.20	33 644.30*	0.00	34 648.70^{+}
	50	9.50	50 851.00	0.36	50 780.70*	0.00	52 419.30^{+}
	75	31.98	114 672.00	1.36	114 605.00*	0.00	120 134.00^{+}
	100	75.94	198 884.00	3.11	198 684.00*	0.00	209 671.00^{+}
*SM*1	10	0.06	150 470.00*	0.00	305 607.00^{+}	0.00	154 277.00
	20	0.45	732 659.00*	0.02	1 073 150.00^{+}	0.00	906 319.00
	30	1.44	1 512 100.00*	0.07	1 768 840.00^{+}	0.00	1 765 760.00
	40	3.45	3 249 080.00*	0.19	3 832 140.00	0.00	4 009 570.00^{+}
	50	6.62	4 509 910.00*	0.35	5 313 940.00	0.00	5 975 470.00^{+}
	75	22.16	11 135 500.00*	1.24	12 238 900.00	0.00	13 105 900.00^{+}
	100	52.29	23 435 700.00*	2.81	24 137 100.00	0.00	32 252 900.00^{+}
*SM*2	10	0.06	142 760.00*	0.00	293 771.00^{+}	0.00	146 447.00
	20	0.45	664 283.00*	0.02	981 248.00^{+}	0.00	778 273.00
	30	1.45	1 208 060.00*	0.08	1 407 020.00	0.00	1 448 110.00^{+}
	40	3.48	2 570 080.00*	0.20	3 008 500.00^{+}	0.00	2 902 690.00
	50	6.68	3 349 290.00*	0.36	3 961 910.00	0.00	4 487 440.00^{+}
	75	22.42	7 109 220.00*	1.26	7 870 520.00	0.00	8 597 210.00^{+}
	100	52.95	13 175 600.00*	2.80	13 673 700.00	0.00	18 327 800.00^{+}
*SM*3	10	0.05	733.99*	0.00	1151.43^{+}	0.00	749.51
	20	0.45	2397.88*	0.02	3188.48^{+}	0.00	2689.12
	30	1.45	4163.81*	0.07	4825.66^{+}	0.00	4718.05
	40	3.47	6914.60*	0.19	8127.66^{+}	0.00	7617.44
	50	6.63	9464.42*	0.35	10 693.70	0.00	11 161.50^{+}
	75	22.22	17 830.90*	1.23	19 591.90	0.00	21 767.20^{+}
	100	52.40	30 896.10*	2.75	31 797.70	0.00	37 760.50^{+}

time distributions. To globally evaluate a node, the partial schedule at that node was first completed according to the SPT rule. Next, constant uptimes and downtimes were inserted and a new schedule that represents an approximate realization was obtained. Job completion times in this new schedule were used to calculate the performance measure of the node instead of simulation. The computational tests in Goren and Sabuncuoglu's correlation study (2008) indicate that $\lambda = 0.6$ performs well. Since the same up- and downtime distributions are used in this study, the same λ value is also used.

To observe the impact of different repair time distributions on the performance of the proposed BS algorithm, the experiments were also conducted with an exponential repair time distribution (with the same mean) instead of gamma.

During our tests, we took the processing time distributions as exponential except for *RM*1. We used a normal distribution for *RM*1 because the SEPT schedule would be already optimal if the processing times were exponentially distributed (see Theorem 1). For *RM*1, variances of the processing times were generated as uniformly distributed over [1, 100]. If a negative processing time value was generated during the simulations, it was simply ignored and generated again.

The simulations (both as a global evaluator in the BS algorithms and as an estimator of the resulting objective function value for all algorithms) during the experiments performed in this section were replicated 100 times instead of ten.

A summary of the results is given in Tables 9 and 10; the best objective function values are marked with an asterix (*) whereas the worst ones are marked with a "+" sign.

As can be seen in Tables 9 and 10, for the *RM*1 performance measure the proposed BS and BS-M1 are competitive. For the case with gamma repair time distribution, the proposed BS generally performs better, whereas BS-M1 performs the best for the case with an exponential repair time distribution.

For all three stability measures, the proposed BS algorithm is significantly better than the corresponding dispatching rule or BS-M1. We observe that BS-M1 gets better with increasing problem sizes. Regardless of the repair time distribution, dispatching rules perform better than BS-M1

Table 11. Dispatching rules, *RM*2, gamma repair times

	Dispatching rule		
Number of jobs	*ATC objective*	*ProATC1 objective*	*ProATC2 objective*
10	857.31	1140.26	856.35
20	2872.70	3672.90	2846.45
30	5875.01	7833.76	5819.89
40	10 354.70	12 943.80	10 252.10
50	15 776.20	20 532.40	15 617.40
75	34 562.20	43 043.50	33 684.00
100	62 468.50	75 501.60	61 303.20

Table 13. Dispatching rules, *RM*2, exponential repair times

	Dispatching rule	
ATC objective	*ProATC1 objective*	*ProATC2 objective*
904.49	1204.02	902.99
3115.63	3906.07	3129.30
6382.19	8546.33	6389.85
11 301.30	14 081.70	11 248.30
17 412.10	22 562.80	17 160.90
37 634.10	47 567.30	37 413.10
69 155.10	84 180.40	67 956.20

for small problems while BS-M1 performs better for larger problems.

We also observe that the differences between the performances of the alternative algorithms for *RM*1 are relatively small compared to the other measures. The reason for such a good performance of SEPT for *RM*1 is that the optimality conditions stated in Theorem 1 are mostly satisfied for *RM*1 (except for stochastic comparability), whereas these conditions are not satisfied due to machine breakdown/repair for other measures and their respective theorems. This indicates that relaxing the stochastic comparability constraint is not as serious as relaxing the constraints on the machine breakdown/repair process. The summary of the results for *RM*2 is given in Tables 11 to 14.

We make three main observations. First, the proactive approach does not always improve the performance of dispatching rules (in particular, ATC in our case) if it is not appropriately used. This is attested to by the better performance of traditional ATC over ProATC1. Our further investigation of this result indicates that how total repair time is distributed is important for the proactive use of dispatching rules. Recall that ProATC1 inserts the repair times for all jobs in proportion to their processing times. ProATC2, however, inserts the repair times by estimating the locations of the machine breakdowns in the sequence. Our results indicate that the latter method (ProATC2) performs significantly better than the former approach (ProATC1) and classical ATC.

Our second main observation is that classical ATC is better than the classical BS for ten-, 20-, 30- and 40-job problems. BS yields better performance than ATC only for large problems. On the other hand, however, simulation-based BS is better than all ATC versions for all problem sizes. This indicates that using simulation as a global evaluation function improves the proposed BS significantly. Nevertheless, we should also note that using simulation as a global evaluation function increases the CPU times exponentially with increasing problem sizes.

In the final observation, we note that the advantage of using the proactive approach becomes more significant for large problem sizes. For example, simulation-based BS yields better results for ten-job problems whereas proactive BS is better for 20 or more job problems. Also, ProATC2 displays a progressively better performance than ATC when the problem size increases. Similarly, we observe that BS-M1 gets better with increasing problem sizes.

In summary, we can conclude that the proposed BS algorithm is quite promising for generating robust or stable schedules. It can also handle computationally intractable cases such as problems with a general machine breakdown/repair process.

Table 12. Comparison of algorithms, *RM*2, gamma repair times

	BS							
	Classical BS		*Simulation-based BS*		*Proactive BS*		*Surrogate BS-M1*	
Number of jobs	*CPU time*	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*
10	0.00	975.64	0.07	787.01	0.07	788.55	0.00	965.75
20	0.04	2960.23	0.57	2682.17	0.56	2671.04	0.04	2866.22
30	0.15	6327.45	1.89	5668.54	1.89	5615.59	0.16	6034.95
40	0.44	10 063.00	4.59	9143.85	4.58	9031.98	0.45	9593.11
50	0.96	15 991.60	9.05	14 482.00	9.05	14 196.40	0.98	15 037.50
75	4.42	32 418.20	31.57	29 676.70	31.54	29 182.60	4.49	30 441.40
100	13.12	56 672.20	77.23	52 260.20	77.14	50 702.90	13.29	52 790.50

Table 14. Comparison of algorithms, *RM*2, exponential repair times

BS							
Classical BS		*Simulation-based BS*		*Proactive BS*		*Surrogate BS-M1*	
CPU time	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*	*CPU time*	*Objective*
0.00	1033.22	0.05	848.80	0.05	852.23	0.00	1021.84
0.04	3163.42	0.46	2858.13	0.45	2843.37	0.04	3056.68
0.15	6933.54	1.52	6192.29	1.52	6126.35	0.15	6597.78
0.44	11 152.00	3.68	10 092.00	3.68	9 917.35	0.44	10 572.50
0.97	17 693.20	7.27	15 994.40	7.26	15 589.20	0.99	16 530.50
4.43	36 327.90	25.49	33 075.00	25.48	32 261.40	4.55	33 712.50
13.13	63 984.20	62.77	58 651.80	62.76	56 361.90	13.52	58 573.90

7. Concluding remarks and future research directions

In this paper, we study proactive scheduling in a single-machine environment with random processing times and random machine breakdowns. We use an expected performance measure as the robustness criterion. We also consider three stability measures. Formal probability theory is used to analyze these measures and some optimality conditions are developed. In this study, we develop an exact algorithm for single-machine scheduling problems with processing time uncertainties. We also develop a BS algorithm as a heuristic to handle cases with machine breakdown/repair.

Minimizing expected total weighted flow time in a single-machine environment subject to random machine breakdowns is known to be NP-hard. Even though the status of the unweighted case (minimizing *RM*1) is unknown, it can be said that the problem is analytically intractable, for it is difficult even to calculate the objective function value of a given solution. For the special case where job processing times are stochastically orderable, Theorem 1 gives the optimal solution.

As for *RM*2, Theorem 3 gives a dominance rule for the case where no breakdowns are present and job processing times are stochastically orderable. Consideration of breakdowns or relaxing the stochastically orderable assumption quickly renders the problem analytically intractable, for it is known that the problem is NP-hard, as stated in Theorem 2.

*SM*1 and *SM*2 are closely related, thus we summarize them together. Sequencing the jobs according to a non-decreasing order of job processing time variances (SVPT) is optimal if no machine breakdowns are present (Theorem 4). If machine breakdowns are included, the SVPT rule is still optimal when the uptimes are exponential and the SVPT sequence coincides with the SEPT sequence (Theorem 5). Relaxing either of these assumptions, i.e., exponential uptimes or coincidence of SEPT and SVPT, the problem becomes analytically intractable. Just as in the case of minimizing *RM*1, even the objective function of a given feasible solution cannot be calculated analytically.

If the processing times are not random variables and the machine uptimes are exponentially distributed, SPT is optimal for minimizing *SM*3 (Theorem 6). Relaxation of either of these assumptions again renders the problem analytically intractable.

To sum up, in this study, we model uncertainty regarding job processing times and machine reliability with known probability distributions. We define several robustness and stability measures. This study contributes to the existing proactive scheduling literature in two ways: first, we identify the analytically tractable cases and we develop an exact algorithm to solve the common problem of minimizing the expected total tardiness using the insights gained while studying these cases. Second, for intractable cases, rather than taking an indirect approach by employing surrogate measures, we estimate the actual measures directly using simulation. The use of simulation in the existing studies may have been avoided because of its anticipated high computational burden. Our computational results, however, indicate that a BS algorithm that employs simulation as a global evaluation function is quite promising and requires reasonable computational times.

We can identify several further research directions. First, the proposed BS algorithm can be extended to more general multi-machine environments. Additionally, the job population in this study is fixed and all jobs are available at time 0. Inclusion of non-zero ready times and dynamic job arrivals will make the approach more applicable to real-life problems.

Second, both robustness and stability are important performance measures for the practitioners. A bicriteria algorithm that can handle both measures is of practical importance. The relationship and the trade-off between robustness and stability can also be analyzed.

Finally, robustness can be measured from a different point of view. For example, the notion of a β-robust schedule is used for the total flow time measure in the literature. A β-robust schedule maximizes the probability of achieving

a system performance less than or equal to a given level T (Daniels and Carillo, 1997). The same concept can be used when the performance measure is total tardiness. Along the same lines, new, easy-to-calculate robustness or stability measures can be developed. The insight gained from this study suggests that it is hard to find an exact method even when we slightly relax the optimality conditions in the theorems developed in Section 3 of this paper. In fact, there are other approaches in the literature that are used when dealing with uncertainty, including scenario planning and modeling with fuzzy numbers. We believe that such approaches could help alleviate the problems encountered in an analytical approach, such as the one taken in this study.

References

Adiri, I., Bruno, J., Frostig, E. and Rinnooy Kan, A.H.G. (1989) Single machine flow-time scheduling with a single breakdown. *Acta Informatica*, **26**(7), 679–696.

Adiri, I., Frostig, E. and Rinnooy Kan, A.H.G. (1991) Scheduling on a single machine with a single breakdown to minimize stochastically the number of tardy jobs. *Naval Research Logistics*, **38**(2), 261–271.

Chu, C. (1992) A branch and bound algorithm to minimize total tardiness with different release times. *Naval Research Logistics*, **39**, 265–283.

Daniels, R. and Kouvelis, P. (1995) Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, **41**(2), 363–376.

Daniels, R.L. and Carillo, J.E. (1997) β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, **29**, 977–985.

Della Croce, F., Tadei, R., Baracco, P. and Grosso, A. (1998) A new decomposition approach for the single machine total tardiness scheduling problem. *Journal of the Operational Research Society*, **49**, 1101–1006.

Du, J. and Leung, T. (1990) Minimizing total tardiness on one machine is NP-hard. *Operations Research*, **15**, 483–495.

Fisher, M.L. (1976) A dual algorithm for the one machine scheduling problem. *Mathematical Programming*, **11**, 229–251.

Goren, S. and Sabuncuoglu, I. (2008) Robustness and stability measures for scheduling: single-machine environment. *IIE Transactions*, **40**(1), 66–83.

Goren, S. and Sabuncuoglu, I. (2009) Generating robust and stable schedules for a single machine environment under random machine breakdowns and processing time variability. Working paper, IE/OR 2009-02, Department of Industrial Engineering, Bilkent University, Ankara.

Herroelen, W. and Leus, E. (2005) Project scheduling under uncertainty: survey and research potentials. *European Journal of Operational Research*, **165**, 289–306.

Kouvelis, P. and Yu, G. (1996) *Robust Discrete Optimization and Its Applications*, Kluwer, Dordrecht, The Netherlands.

Law, A.M. and Kelton, W.D. (2000) *Simulation Modeling and Analysis*, third edition, McGraw-Hill, Singapore.

Leon, V.J., Wu, S.D. and Storer, R.H. (1994) Robustness measures and robust scheduling for job shops. *IIE Transactions*, **26**(5), 32–43.

Leung, J.Y.-T. and Pinedo, M. (2004) A note on scheduling parallel machines subject to breakdown and repair. *Naval Research Logistics*, **51**(1), 60–71.

Li, W., Braun, W.J. and Zhao, Y.Q. (1998) Stochastic scheduling on a repairable machine with Erlang uptime distribution. *Advances in Applied Probability*, **30**(4), 1073–1088.

Li, W. and Glazebrook, K.D. (1998) On stochastic machine scheduling with general distributional assumptions. *European Journal of Operational Research*, **105**, 525–536.

Mehta, S.V. and Uzsoy, R. (1998) Predictable scheduling of a job shop subject to breakdowns. *IEEE Transactions on Robotics and Automation*, **14**(3), 365–378.

Mehta, S.V. and Uzsoy, R. (1999) Predictable scheduling of a single machine subject to breakdowns. *International Journal of Computer-Integrated Manufacturing*, **12**, 15–38.

Nelson, R.T., Holloway, C.A. and Wong, R.M. (1977) Centralized scheduling and priority implementation heuristics for a dynamic job shop model with due dates and variable processing times. *AIIE Transactions*, **19**, 227–241.

O'Donovan, R., Uzsoy, R. and McKay, K.N. (1999) Predictable scheduling of a single machine with breakdowns and sensitive jobs. *International Journal of Production Research*, **37**(18), 4217–4233.

Pinedo, M. (2002) *Scheduling Theory, Algorithms, and Systems*, Prentice Hall, Upper Saddle River, NJ.

Ross, S.M. (1983) *Stochastic Processes*, Wiley, Somerset, NJ.

Ross, S.M. (1993) *Introduction to Probability Models*, Academic Press, Boston, MA.

Sabuncuoglu, I. and Goren, S. (2009) Hedging production schedules against uncertainty in manufacturing environment with a review of robustness and stability research. Working paper, IE/OR 2009-03, Department of Industrial Engineering, Bilkent University, Ankara.

Sabuncuoglu, I. and Karabuk, S. (1998) A beam search based algorithm and evaluation of scheduling approaches. *IIE Transactions*, **30**(2), 179–191.

Sotskov, Y., Sotskova, N.Y. and Werner, F. (1997) Stability of an optimal schedule in a job shop. *Omega: The International Journal of Management Science*, **25**(4), 397–414.

Vepsalainen, A.P.J. and Morton, T.E. (1987) Priority rules for job shops with weighted tardiness costs. *Management Science*, **33**, 1035–1047.

Wu, S.D., Byeon, E. and Storer, R.H. (1999) A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness. *Operations Research*, **47**(1), 113–124.

Wu, S.D., Storer, R.H. and Chang, P. (1993) One-machine rescheduling heuristics with efficiency and stability as criteria. *Computers Opreations Research*, **20**(1), 1–14.

Appendix

Proof of Theorem 5 for *SM*1. The proof is by contradiction. Let S be an optimal sequence but assume that there exists a pair of adjacent jobs i and j such that $E[X_i] > E[X_j]$ and job j succeeds job i in S. Now consider a sequence $S\prime$, obtained from S by swapping the positions of jobs i and j. We compare $SM1(S)$ and $SM1(S')$. We may ignore the jobs other than i and j in this comparison, since nothing changes for them. Let their contribution to the objective function be c. Let BS_i denote the index set of jobs that precedes job i in S. Let $A_k = Y_k - E[X_k]$ for each job index k. We have:

$$SM1(S) = E\left[\left(\sum_{m\in BS_i} Y_m + Y_i - E\left[\sum_{m\in BS_i} X_m\right] - E[X_i]\right)^2\right] + E\left[\left(\sum_{m\in BS_i} Y_m + Y_i + Y_j - E\left[\sum_{m\in BS_i} X_m\right]\right.\right.$$

$$- E[X_i] - E[X_j]\Bigg)^2\Bigg] + c$$
$$= E\Bigg[\Bigg(A_i + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] + E\Bigg[\Bigg(A_i + A_j + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] + c.$$

Similarly,

$$SM1(S') = E\Bigg[\Bigg(A_j + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] + E\Bigg[\Bigg(A_i + A_j + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] + c.$$

Then,

$$\begin{aligned} &SM1(S) - SM1(S') \\ &= E\Bigg[\Bigg(A_i + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] - E\Bigg[\Bigg(A_j + \sum_{m\in BS_i} A_m\Bigg)^2\Bigg] \\ &= E\Bigg[A_i^2 + \Bigg(\sum_{m\in BS_i} A_m\Bigg)^2 + 2A_i \sum_{m\in BS_i} A_m\Bigg] \\ &\quad - E\Bigg[A_j^2 + \Bigg(\sum_{m\in BS_i} A_m\Bigg)^2 + 2A_j \sum_{m\in BS_i} A_m\Bigg] \\ &= E\big[A_i^2\big] - E\big[A_j^2\big] + 2E[A_i - A_j]E\Bigg[\sum_{m\in BS_i} A_m\Bigg]. \end{aligned}$$

The last line is obtained by using the fact that the A_ks are independent, since the X_ks and Y_ks are independent. Note that $E[A_i] = E[Y_i] - E[X_i] = \lambda r\, E[X_i]$ and

$$\begin{aligned} E\big[A_i^2\big] &= E\big[(Y_i - E[X_i])^2\big] \\ &= E\big[Y_i^2 + (E[X_i])^2 + 2Y_i E[X_i]\big] \\ &= E\big[Y_i^2\big] + (E[X_i])^2 + 2E[Y_i]E[X_i] \\ &= \mathrm{Var}[Y_i] + (E[Y_i])^2 + (E[X_i])^2 + 2E[Y_i]E[X_i] \\ &= \lambda(v + r^2)E[X_i] + (1 + \lambda^2 r^2)\mathrm{Var}[X_i] \\ &\quad + (1 + \lambda r)^2(E[X_i])^2 + (E[X_i])^2 + 2(1 + \lambda r)(E[X_i])^2 \\ &= \lambda(v + r^2)E[X_i] + (1 + \lambda^2 r^2)\mathrm{Var}[X_i] \\ &\quad + (\lambda^2 r^2 + 4\lambda r + 4)(E[X_i])^2. \end{aligned}$$

Then we have:

$$\begin{aligned} &SM1(S) - SM1(S') \\ &= \lambda(v + r^2)(E[X_i] - E[X_j]) + (1 + \lambda^2 r^2)(\mathrm{Var}[X_i] \\ &\quad - \mathrm{Var}[X_i]) + (\lambda^2 r^2 + 4\lambda r + 4)((E[X_i])^2 - (E[X_j])^2) \\ &\quad + 2\lambda r(E[X_i] - E[X_j])E\Bigg[\sum_{m\in BS_i} A_m\Bigg] \end{aligned}$$

Since $E[X_i] > E[X_j]$ and $\mathrm{Var}[X_i] \geq \mathrm{Var}[X_j]$ this difference is strictly positive, which contradicts with the optimality of S.

The proof can be done with the same interchange argument for $SM2$. ■

Biographies

Selcuk Goren is a Ph.D. candidate in the Department of Industrial Engineering at Bilkent University. He received his B.S. and M.S. degrees from Bilkent University. His primary research interest is scheduling under uncertainty.

Ihsan Sabuncuoglu is a Professor of Industrial Engineering at Bilkent University. He received B.S. and M.S. degrees in Industrial Engineering from the Middle East Technical University and a Ph.D. degree in Industrial Engineering from Wichita State University. He teaches and conducts research in the areas of scheduling, production management, simulation and manufacturing systems. He has published papers in *IIE Transactions*, *Decision Sciences*, *Simulation*, *International Journal of Production Research*, *International Journal of Flexible Manufacturing Systems*, *International Journal of Computer Integrated Manufacturing*, *European Journal of Operational Research*, *Production Planning and Control*, *Journal of the Operational Research Society*, *Computers & Operations Research*, *Computers & Industrial Engineering*, *OMEGA, Journal of Intelligent Manufacturing* and *International Journal of Production Economics*. He is on the Editorial board of *International Journal of Operations and Quantitative Management* and also the *Journal of Operations Management*. He is an associate member of Institute of Industrial Engineering, Institute of Simulation and Institute of Operations Research and Management Science.