IIE Transactions (2000) **32**, 421–432

Robust scheduling of a two-machine flow shop with uncertain processing times

PANOS KOUVELIS[1], RICHARD L. DANIELS[2] and GEORGE VAIRAKTARAKIS[3]

[1]*Olin School of Business, Washington University, St. Louis, MO 63130-4899, USA*
E-mail: Kouvelis@wuolin.wustl.edu
[2]*DuPree College of Management, Georgia Institute of Technology, Atlanta, GA 30332-0520, USA*
E-mail: rich.daniels@mgt.gatech.edu
[3]*Weatherhead School of Management, Case Western Reserve University, Cleveland, OH 44106-7235, USA*
E-mail: gxvs@po.cwru.edu

Received October 1997 and accepted October 1998

This paper focuses on manufacturing environments where job processing times are uncertain. In these settings, scheduling decision makers are exposed to the risk that an optimal schedule with respect to a deterministic or stochastic model will perform poorly when evaluated relative to actual processing times. Since the quality of scheduling decisions is frequently judged as if processing times were known *a priori*, robust scheduling, i.e., determining a schedule whose performance (compared to the associated optimal schedule) is relatively insensitive to the potential realizations of job processing times, provides a reasonable mechanism for hedging against the prevailing processing time uncertainty. In this paper we focus on a two-machine flow shop environment in which the processing times of jobs are uncertain and the performance measure of interest is system makespan. We present a measure of schedule robustness that explicitly considers the risk of poor system performance over all potential realizations of job processing times. We discuss two alternative frameworks for structuring processing time uncertainty. For each case, we define the robust scheduling problem, establish problem complexity, discuss properties of robust schedules, and develop exact and heuristic solution approaches. Computational results indicate that robust schedules provide effective hedges against processing time uncertainty while maintaining excellent expected makespan performance.

1. Introduction

We consider a two-stage flow shop where the processing times of individual jobs on each machine are uncertain, and the performance measure of interest is system makespan. For this type of problem, the stochastic scheduling literature has generally assumed independent and known processing time distributions for individual jobs, and an objective of optimizing expected system performance [1–6]. In contrast, we develop approaches for generating production schedules that focus on the worst-case deviation (over all potential processing time scenarios) between actual system performance and the optimal performance that could be achieved with perfect processing time information. This *robust* scheduling approach reflects the concerns of risk-averse decision makers who may be more interested in hedging against poor system performance for some realizations of job processing times than in optimizing expected system performance over all potential scenarios. A similar robust scheduling approach was developed for a single-machine environment by Daniels and Kouvelis [7]. Other robust decision-making formulations have been presented by Rosenblatt and Lee [8], Kouvelis *et al.* [9], and Mulvey *et al.* [10].

The paper is organized as follows. In Section 2, we define a measure of schedule robustness, define the robust scheduling problem, and establish problem complexity. We also discuss two ways in which processing time uncertainty can be structured. The first approach assumes a set of discrete processing time scenarios, each of which specifies the processing time of each job on each machine under that scenario. The second approach assumes a set of independent processing time intervals, which define the minimum and maximum processing time that can be realized for each job on each machine. In Section 3, we present branch-and-bound solution approaches for the robust two-machine flow shop scheduling problem for both the discrete processing time scenario case and the processing time interval case. Heuristic approaches for the problems are developed in Section 4. Section 5 reports the computational performance of the procedures, and shows that robust schedules provide optimal hedges against the prevailing processing time uncertainty while maintaining excellent expected makespan performance.

Section 6 concludes with a summary; proofs of all results can be found in the Appendix.

2. Formulation of the robust two-machine flow shop scheduling problem

Consider a set $\{1,2,\ldots,n\}$ of n independent jobs that require processing on each of two machines. The flow of work through the shop is unidirectional, so that processing of any job on machine 2 cannot commence until that job completes processing on machine 1. We assume that the processing times of individual jobs are uncertain due to, e.g., incomplete/unreliable information or unavoidable stochastic variability. This processing time uncertainty is described through a set of processing time scenarios Λ. Each scenario $\lambda \in \Lambda$ represents a unique set of processing times for each job on each machine, which can be realized with some positive probability. Let p_{ij}^{λ} denote the processing time of job i on machine j in scenario λ, and $\mathbf{P}^{\lambda} = \{p_{ij}^{\lambda} : i = 1,2,\ldots,n,\ j = 1,2\}$ the vector of processing times associated with scenario λ.

Let Ω represent the set of all permutation sequences constructed from the n jobs. Let $\sigma = \{\sigma(1),\sigma(2),\ldots,\sigma(n)\}$ denote a given sequence in Ω, and $\sigma_{\lambda}^{*} = \{\sigma_{\lambda}^{*}(1),\sigma_{\lambda}^{*}(2),\ldots,\sigma_{\lambda}^{*}(n)\}$ the optimal sequence given processing time scenario $\lambda \in \Lambda$ (σ_{λ}^{*} can be easily constructed using Johnson's algorithm [11]), with $\sigma(k)$ and $\sigma_{\lambda}^{*}(k)$ representing the job that occupies position k in sequences σ and σ_{λ}^{*}, respectively. Let $\varphi(\sigma,\mathbf{P}^{\lambda})$ denote the makespan of schedule $\sigma \in \Omega$ given processing time scenario λ. Then the optimal schedule σ_{λ}^{*} given processing time vector $\mathbf{P}^{\lambda}$ must satisfy:

$$z^{\lambda} = \varphi(\sigma_{\lambda}^{*},\mathbf{P}^{\lambda}) = \min_{\sigma\in\Omega} \varphi(\sigma,\mathbf{P}^{\lambda}). \tag{1}$$

The objective is to minimize the maximum possible regret associated with a schedule. We measure regret for a given schedule and set of processing times as the absolute difference between the makespan of the schedule for that scenario and the makespan of the optimal Johnson schedule for that scenario. Thus, the *absolute deviation robust* schedule, σ_R, can be identified such that:

$$\begin{aligned}&\max_{\lambda\in\Lambda}\{\varphi(\sigma_R,\mathbf{P}^{\lambda}) - \varphi(\sigma_{\lambda}^{*},\mathbf{P}^{\lambda})\}\\ &\quad = \min_{\sigma\in\Omega}\max_{\lambda\in\Lambda}\{\varphi(\sigma,\mathbf{P}^{\lambda}) - \varphi(\sigma_{\lambda}^{*},\mathbf{P}^{\lambda})\},\end{aligned} \tag{2}$$

and the problem of determining the schedule that satisfies (2) is referred to as the Absolute Deviation Robust Scheduling (ADRS) problem.

To determine σ_R, define the following decision variables:

$$x_{ik} = \begin{cases} 1, & \text{if } \sigma(k) = i,\\ 0, & \text{otherwise.}\end{cases}$$

B_k^{λ} = start time of job $\sigma(k)$ on machine 2 given processing time scenario λ.

The ADRS problem for the two-machine flow shop with system makespan the performance measure of interest can then be formulated as:

$$\text{Minimize } y \tag{3}$$

s.t.

$$\sum_{i=1}^{n} p_{i2}^{\lambda} x_{in} + B_n^{\lambda} \le y + z^{\lambda}, \quad \lambda \in \Lambda, \tag{4}$$

$$\sum_{i=1}^{n}\sum_{\ell=1}^{k} p_{i1}^{\lambda} x_{i\ell} \le B_k^{\lambda}, \quad k = 1,2,\ldots,n;\ \lambda \in \Lambda, \tag{5}$$

$$B_k^{\lambda} + \sum_{i=1}^{n} p_{i2}^{\lambda} x_{ik} \le B_{k+1}^{\lambda},\ k = 1,2,\ldots,n-1;\ \lambda \in \Lambda, \tag{6}$$

$$\sum_{i=1}^{n} x_{ik} = 1, \quad k = 1,2,\ldots,n, \tag{7}$$

$$\sum_{k=1}^{n} x_{ik} = 1, \quad i = 1,2,\ldots,n, \tag{8}$$

$$x_{ik} \in \{0,1\}, \quad i,k = 1,2,\ldots,n. \tag{9}$$

Thus, objective (3) and constraints (4) ensure that the optimal schedule conforms to the definition of the absolute deviation robust schedule given in (2). Constraints (5)–(9) guarantee that the robust schedule is feasible, and that system makespan is appropriately calculated. The complexity of the ADRS problem is established by the following result.

Theorem 1. *The two-machine ADRS problem is NP-hard.*

The processing time uncertainty that underlies the ADRS problems can be structured in at least two ways. The *discrete scenario* case assumes a set of discrete processing time scenarios, each of which specifies the processing time of each job on each machine under that scenario. Alternatively, processing time variability can be modeled by specifying that the processing time of job i on machine j may fall within the range bounded by $\underline{p}_{ij}$ and $\bar{p}_{ij}$. In most cases, $\underline{p}_{ij}$ and $\bar{p}_{ij}$ are easier to specify than the entire probability distribution. While the alternative structure implies an infinite number of scenarios, the problem formulation represented by expression (2) is valid for this *continuous processing time interval* case.

3. Branch-and-bound algorithm for the ADRS problem

In this section we develop a branch-and-bound algorithm for the ADRS problem, and apply the procedure to the robust scheduling problem with discrete processing time scenarios, and to the problem with processing time intervals. In each case, the algorithm utilizes a branching scheme that systematically augments partial schedules to

capture all feasible assignments of jobs to sequence positions, with level k of the tree corresponding to jobs occupying the kth position in alternative partial sequences. Partial sequences that violate dominance conditions specified for the problem are immediately eliminated from consideration. Partial sequences are also fathomed when the associated lower bound exceeds the best available upper bound. Complete sequences not excluded by the bounding process are evaluated using a procedure that determines the worst-case absolute deviation from optimality for the associated schedule.

Thus, the process used to determine the worst-case performance of alternative sequences, along with the lower bounds and dominance results established for the problem, characterizes how the branch-and-bound algorithm can be applied to identify robust schedules for problem instances with discrete processing time scenarios, and for the processing time interval case. These procedural elements are discussed in detail in the following sections.

3.1. *Discrete processing time scenarios*

For the discrete scenario case, the worst-case scenario for any given sequence can be determined using a process (referred to as Procedure Worst-Case [Discrete Scenarios]) that requires $|\Lambda|$ iterations, where $|\Lambda|$ represents the number of processing time scenarios. For each $\lambda \in \Lambda$, the makespan of the given sequence is computed in $O(n)$ time, and the corresponding optimal makespan is determined in $O(n \log n)$ time using Johnson's algorithm. If the difference between the two makespans exceeds the largest difference encountered in iterations 1 through $\lambda - 1$, the new value and associated scenario are retained. After $|\Lambda|$ iterations, the stored value represents the worst-case absolute deviation from optimality for the given sequence, and the associated set of processing times represents the worst-case scenario for that sequence. By determining the set of optimal makespans exactly once in $O(n|\Lambda| \log n)$ time (external to Procedure Worst-Case [Discrete Scenarios]), the complexity of the worst-case evaluation process is $O(|\Lambda|n)$.

To obtain a lower bound on the worst-case deviation from optimality for any partial schedule, we utilize a bounding process that requires $|\Lambda|$ iterations. To illustrate the process, consider a partial sequence $1 - 2 - \cdots - \ell$, with the sequence of the remaining $n - \ell$ jobs as yet unknown. At iteration λ of the bounding process, jobs are assigned their processing times in scenario λ, *i.e.*, $p_{ij} = p_{ij}^{\lambda}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2$. The $n - \ell$ unassigned jobs are then arranged in Johnson order with respect to the p_{ij}^{λ} and appended to the current partial schedule. The difference between the makespan of this schedule and the corresponding optimal makespan (both computed with respect to the p_{ij}^{λ}) represents a lower bound on the absolute deviation from optimality that can be realized given the partial sequence and processing time scenario λ. The maximum lower bound obtained by repeating this process for each $\lambda \in \Lambda$ represents the minimum worst-case deviation from optimality that can be realized for any schedule starting with the given partial sequence. Again, if the Johnson sequence associated with each processing time scenario is determined external to the bounding process, then the complexity of the procedure is $O(|\Lambda|n)$.

The following dominance properties are also employed to enhance the computational efficiency of the branch-and-bound procedure.

Property 1. *If for two jobs i and h, $p_{i1}^{\lambda} \le p_{h1}^{\lambda}$ and $p_{i2}^{\lambda} \ge p_{h2}^{\lambda}$ for all $\lambda \in \Lambda$, then there exists an absolute deviation robust schedule in which job i precedes job h.*

Property 2. *If for two jobs i and h, $\min\{p_{i1}^{\lambda}, p_{h2}^{\lambda}\} \le \min\{p_{h1}^{\lambda}, p_{i2}^{\lambda}\}$ for all $\lambda \in \Lambda$, then the absolute deviation robust schedule can be determined without explicitly considering schedules in which job h immediately precedes job i.*

These properties are similar to the dominance results presented in the next section for the continuous processing time interval case, and as a result, the proofs of Properties 1 and 2 are nearly identical to those given in the Appendix for Propositions 1 and 2.

3.2. *Continuous processing time intervals*

We first address the problem of determining the worst-case absolute deviation scenario for a given sequence from the infinite set of processing time scenarios obtained when the variability in job processing times is captured by specifying for each job i and machine j a range $[\underline{p}_{ij}, \bar{p}_{ij}]$ of realizable times. The worst-case absolute deviation scenario for sequence σ, λ_0^{σ}, must satisfy:

$$\varphi(\sigma, \mathbf{P}^{\lambda_0^{\sigma}}) - \varphi(\sigma_{\lambda_0^{\sigma}}^{*}, \mathbf{P}^{\lambda_0^{\sigma}}) = \max_{\lambda : \underline{p}_{ij} \le p_{ij}^{\lambda} \le \bar{p}_{ij}, \forall i,j} \left[\varphi(\sigma, \mathbf{P}^{\lambda}) - \varphi(\sigma_{\lambda}^{*}, \mathbf{P}^{\lambda})\right]. \quad (3)$$

Let processing time scenario λ be defined as an *extreme point scenario* if for each job $i = 1, 2, \ldots, n$, and each machine $j = 1, 2$, either $p_{ij}^{\lambda} = \underline{p}_{ij}$ or $p_{ij}^{\lambda} = \bar{p}_{ij}$. The following result demonstrates that the worst-case absolute deviation scenario λ_0^{σ} for any sequence σ is an extreme point scenario, and specifies a property that allows the appropriate extreme value to be directly determined. For a given sequence σ and specific processing time scenario λ, let job $\sigma(k)$ be a *critical job* if processing of job $\sigma(k)$ on machine 2 commences as soon as this job is completed on machine 1. Job $\sigma(i_0)$ is the *last critical job* if for the index set of critical jobs $C_{\lambda} = \{k : \sigma(k) \text{ is a critical job}\}$ for sequence σ and scenario λ, $i_0 \in C_{\lambda}$ and $i_0 \ge k$ for any $k \in C_{\lambda}$. C_{λ} is nonempty, since otherwise for any sequence σ and scenario λ, at least one machine-2 task could be shifted to an earlier time period, thus reducing makespan $\varphi(\sigma, \mathbf{P}^{\lambda})$.

Theorem 2.

(a) *For any sequence σ and given the makespan performance criterion for the two-machine flow shop, the worst-case absolute deviation scenario λ_0^σ belongs to the set of extreme point scenarios.*

(b) *For the worst-case absolute deviation scenario λ_0^σ for sequence σ:*

$$p_{ij}^{\lambda_0^\sigma} = \begin{cases} \bar{p}_{ij}, & \text{if } (i \in V_1^\sigma \cup \{\sigma(i_0)\} \text{ and } j = 1) \\ & \text{or } (i \in V_2^\sigma \cup \{\sigma(i_0)\} \text{ and } j = 2), \\ \underline{p}_{ij}, & \text{if } (i \in V_2^\sigma \text{ and} j = 1) \\ & \text{or } (i \in V_1^\sigma \text{ and } j = 2), \end{cases}$$

where $\sigma(i_0)$ is the last critical job in sequence σ for processing time scenario λ_0^σ, and V_1^σ and V_2^σ are defined as $V_1^\sigma = \{\sigma(k) : k < i_0\}$ and $V_2^\sigma = \{\sigma(k) : k > i_0\}$, respectively.

The worst-case scenario for a given sequence (and the associated worst-case deviation between the makespan of the sequence and the corresponding optimal makespan) is determined using a process (referred to as Procedure Worst-Case [Processing Time Intevals]) that requires n iterations. To illustrate, consider sequence $1 - 2 - \cdots - n$. For iteration k of the procedure, the job occupying position k in sequence (i.e., job k) is assumed to be the last critical job, and the processing times of all jobs are determined using Theorem 2(b). Thus, the processing time of job k on each machine is set to its largest value, i.e., $p_{k1} = \bar{p}_{k1}$ and $p_{k2} = \bar{p}_{k2}$, jobs in front of job k are assigned their largest machine 1 processing time and their smallest machine 2 processing time, i.e., $p_{i1} = \bar{p}_{i1}$ and $p_{i2} = \underline{p}_{i2}$ for $i = 1, 2, \ldots, k-1$, and jobs positioned after job k are assigned their smallest machine 1 processing time and their largest machine 2 processing time, $p_{i1} = \underline{p}_{i1}$ and $p_{i2} = \bar{p}_{i2}$ for $i = k+1, k+2, \ldots, n$. Given this processing time scenario, the makespan of the given sequence is computed and compared to the corresponding optimal makespan, determined by applying the Johnson algorithm to the above set of processing times. If the deviation between the two makespan values is greater than the largest deviation encountered in iterations 1 through $k-1$, the new value and associated scenario are retained. After n iterations, this process yields the worst-case absolute deviation from optimality for the given sequence, and the associated set of processing times represents the worst-case scenario for that sequence.

The complexity of Procedure Worst-Case [Processing Time Intervals] is $O(n^2 \log n)$ because we apply the Johnson algorithm n times, once for every trial position of the last critical job k. However, the complexity can be improved to $O(n^2)$ based on the following observations. Let λ_k and λ_{k+1} be the extreme point scenarios of the form described in Theorem 2(b) when the last critical job is in position $\sigma(k)$ and $\sigma(k+1)$, respectively. Scenarios λ_k and λ_{k+1} differ only in processing times $p_{k+1,1}$ and $p_{k,2}$; more specifically, $p_{k+1,1}^{\lambda_k} = \underline{p}_{k+1,1}$ while $p_{k+1,1}^{\lambda_{k+1}} = \bar{p}_{k+1,1}$, and $p_{k,2}^{\lambda_k} = \bar{p}_{k,2}$ while $p_{k,2}^{\lambda_{k+1}} = \underline{p}_{k,2}$. According to Theorem 2(b), all other tasks have the same processing time in both λ_k and λ_{k+1}. Hence, the Johnson order of jobs $1 - 2 - \cdots - k - 1 - k + 2 - \cdots - n$ with respect to scenario λ_k is the same as that associated with scenario λ_{k+1}. Schedule $\sigma_{\lambda_{k+1}}^\star$ can be found from $\sigma_{\lambda_k}^\star$ by: (i) eliminating jobs k and $k+1$ from $\sigma_{\lambda_k}^\star$ (let σ_s be the resulting subsequence); (ii) updating the processing times of jobs k and $k+1$, i.e., $p_{k+1,1}^{\lambda_{k+1}} = \bar{p}_{k+1,1}$ and $p_{k,2}^{\lambda_{k+1}} = \underline{p}_{k,2}$; and (iii) applying the Johnson rule to reinsert jobs k and $k+1$ in σ_s. Steps (i)–(iii) of the revised procedure only require $O(n)$. Considering all n trial positions for the last critical job, the complexity of Procedure Worst-Case[Processing Time Intervals] is thus $O(n^2)$.

A lower bound for a given partial sequence can be developed using a process similar to Procedure Worst-Case [Processing Time Intervals]. To illustrate, consider partial sequence $1 - 2 - \cdots - \ell$, with the scheduling of the remaining $n - \ell$ jobs as yet unknown. The first stage of the bounding process requires ℓ iterations. At iteration k of this stage, the job occupying position k in sequence (i.e., job k) is assumed to be the last critical job, and the processing times of the jobs are again determined using Theorem 2(b). Thus, the processing time of job k on each machine is set to its largest value, $p_{k1} = \bar{p}_{k1}$ and $p_{k2} = \bar{p}_{k2}$, jobs in front of job k are assigned their largest machine 1 processing time and their smallest machine 2 processing time, $p_{i1} = \bar{p}_{i1}$ and $p_{i2} = \underline{p}_{i2}$ for $i = 1, 2, \ldots, k-1$, and jobs positioned after job k (including the $n - \ell$ unassigned jobs) are assigned their smallest machine 1 processing time and their largest machine 2 processing time, $p_{i1} = \underline{p}_{i1}$ and $p_{i2} = \bar{p}_{i2}$ for $i = k+1, k+2, \ldots, n$. Given this processing time scenario and assuming that job k is the last critical job in the final schedule, we can compute a lower bound on the makespan associated with the given partial sequence, $LB = \sum_{i=1}^{k} \bar{p}_{i1} + \sum_{i=k}^{n} \bar{p}_{i2}$, and determine the corresponding optimal sequence and makespan for this scenario using Johnson's algorithm. If the deviation between the two makespan values is greater than the largest deviation encountered in iterations 1 through $k-1$, the new value is retained. After ℓ iterations, the procedure yields a first lower bound on the worst-case absolute deviation from optimality for the given partial sequence. As above, the complexity of this stage of the process is $O(n^2)$.

The rationale behind the remaining two stages of the bounding process is that we can easily evaluate the impact of each of the unassigned jobs being the last critical job while occupying either position $\ell + 1$ or position n in the final schedule. Suppose unassigned job k is the last critical job while occupying position $\ell + 1$ in sequence. Then job k should have its processing time on each machine set to its largest value, i.e., $p_{k1} = \bar{p}_{k1}$ and $p_{k2} = \bar{p}_{k2}$, the jobs in the partial sequence should all be assigned their largest machine 1 processing time and their smallest machine 2 processing time, $p_{i1} = \bar{p}_{i1}$ and $p_{i2} = \underline{p}_{i2}$ for $i = 1, 2, \ldots, \ell$,

and the remaining unassigned jobs should be assigned their smallest machine 1 processing time and their largest machine 2 processing time, $p_{i1} = \underline{p}_{i1}$ and $p_{i2} = \bar{p}_{i2}$ for $i > \ell$, $i \neq k$. Computing a lower bound on the makespan associated with the given partial sequence, $LB' = \bar{p}_{k1} + \sum_{i=1}^{\ell} \bar{p}_{i1} + \sum_{i>\ell}^{n} \bar{p}_{i2}$, and the corresponding optimal makespan using the Johnson algorithm, we obtain an *upper* bound on the worst-case performance, since job k need not necessarily be either the last critical job or occupy position $\ell + 1$ in sequence. However, if we repeat the process for each of the $n - \ell$ unassigned jobs, the minimum difference represents a lower bound on the worst-case absolute deviation from optimality associated with the given partial sequence, since one unassigned job must ultimately occupy position $\ell + 1$.

Similar reasoning can be applied to measure the impact of each unassigned job being the last critical job while occupying position n in sequence. Consider unassigned job k. If k is the last critical job and occupies position n in sequence, then job k should again have its processing time on each machine set to its largest value, i.e., $p_{k1} = \bar{p}_{k1}$ and $p_{k2} = \bar{p}_{k2}$. All $n - 1$ remaining jobs should be assigned their largest machine 1 processing time and their smallest machine 2 processing time, $p_{i1} = \bar{p}_{i1}$ and $p_{i2} = \underline{p}_{i2}$ for $i < n$, $i \neq k$. Computing a lower bound on the makespan associated with the given partial sequence, $LB'' = \bar{p}_{k2} + \sum_{i=1}^{n} \bar{p}_{i1}$, and the corresponding optimal makespan using the Johnson algorithm, we again obtain an *upper* bound on the worst-case performance, since job k need not necessarily be either the last critical job or occupy position n in sequence. However, if we repeat the process for each of the $n - \ell$ unassigned jobs, the minimum difference represents a lower bound on the worst-case absolute deviation from optimality associated with the given partial sequence, since one unassigned job must ultimately occupy position n. Again, the complexity of each of these two stages of the process is $O(n^2)$.

The following two dominance properties can be used to establish the relative position of pairs of jobs in the robust schedule.

Proposition 1. *If for two jobs i and h, $\bar{p}_{i1} \leq \underline{p}_{h1}$ and $\underline{p}_{i2} \geq \bar{p}_{h2}$, then there exists an absolute robust schedule in which job i precedes job h.*

Proposition 2. *If for two jobs i and h, $\min\{\bar{p}_{i1}, \bar{p}_{h2}\} \leq \min\{\underline{p}_{h1}, \underline{p}_{i2}\}$, then the absolute deviation robust schedule can be determined without explicitly considering schedules in which job h immediately precedes job i.*

Observe that Proposition 2 generalizes Johnson's property for an optimal solution to the deterministic two-machine flow shop scheduling problem.

It is important to note that the robust schedule is a direct byproduct of the smallest and largest processing times that can be realized for each job on each machine. A robust scheduling approach thus requires a decision maker to specify processing time ranges where the subjective probability of realizing an extreme processing time is sufficient to justify explicit consideration of the associated risk (captured by the deviation between actual and optimal makespan performance) borne if that extreme processing time is realized. Processing time ranges in which the extremes are realized with infinitesimal probability reflect a willingness to hedge against extremely unlikely outcomes; otherwise, tighter processing time ranges whose extremes correspond to more likely realizations should be used. In either case, the process outlined in this section yields an appropriate robust schedule that minimizes the worst-case deviation from optimal makespan performance for the processing time ranges specified. Our computational experience indicates that small to moderate increases in the minimum deviation from optimality are realized when processing time ranges are widened by up to 50%, and that the identity of the robust schedule changes very little as processing time ranges are enlarged. Thus, a decision maker can have considerable confidence in the robust schedule even when processing time ranges can be only specified with some uncertainty.

4. Heuristic solution approaches for the ADRS problem

The complexity of the ADRS Problem motivates the development of heuristics capable of yielding close approximations of the robust schedule with little computational effort. Heuristic solutions also represent initial upper bounds in the branch-and-bound algorithms described in the previous section. Heuristics for the discrete processing time scenario case and for the processing time intervals cases are detailed in this section, and the procedures are computationally tested in Section 5.

4.1. *Discrete processing time scenarios*

The rationale behind the heuristic approach for the discrete processing time scenario case is that the robust schedule will likely be structurally similar to at least one of the schedules that is optimal with respect to an individual processing time scenario. The heuristic procedure requires $|\Lambda|$ iterations. In iteration λ, processing times are set to the values associated with scenario λ, i.e., $p_{ij} = p_{ij}^{\lambda}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2$. The Johnson sequence for this set of processing times is constructed, and its worst-case performance determined using Procedure Worst-Case [Discrete Scenarios]. We then seek to improve the initial sequence by considering the $n(n-1)$ insertions and $n(n-1)/2$ pairwise interchanges that can be derived from this schedule. The worst-case deviation from optimality for each generated sequence is then determined using

Procedure Worst-Case [Discrete Scenarios]. If no improvement over the performance of the incumbent sequence is realized, iteration λ of the process is completed, and the procedure advances to scenario $\lambda + 1$. Otherwise, the alternative sequence yielding the lowest worst-case deviation from optimality becomes the incumbent sequence, and the process of generating and evaluating the worst-case performance of sequences in the neighborhood of the incumbent sequence is repeated. The process continues until the set of alternative sequences yields no improvement over the incumbent sequence. Upon completion of iteration λ, the worst-case performance of the incumbent sequence is compared with that of the best sequence encountered in iterations 1 through $\lambda - 1$, and the best sequence is retained. After $|\Lambda|$ iterations, the incumbent sequence represents an approximation of the schedule that minimizes the worst-case absolute deviation from optimality. The procedure is detailed below.

Procedure Improve [Discrete Scenarios]

Input. The set of processing time scenarios, $\Lambda = \{\mathbf{P}^\lambda : \lambda \in \Lambda\}$, where $\mathbf{P}^\lambda = \{p_{ij}^\lambda\}$ for $i = 1, 2, \ldots, n$ and $j = 1, 2$.

Output. Heuristic sequence, σ_h, that provides an Upper Bound (UB) on the minimum worst-case absolute deviation from optimality.

Step 1. Set $\lambda = 0$, $S = \emptyset$, and $UB = \infty$.

Step 2. Set $\lambda = \lambda + 1$. Construct sequence σ_h^λ by arranging the jobs in Johnson order with respect to the processing times associated with scenario λ. Determine the worst-case absolute deviation from optimality for sequence σ_h^λ (denoted by $d_{\sigma_h^\lambda}$) using Procedure Worst-Case [Discrete Scenarios], and set $UB^\lambda = d_{\sigma_h^\lambda}$. If $\sigma_h^\lambda \in S$, repeat Step 2; otherwise, set $S = S \cup \{\sigma_h^\lambda\}$, $UB_{\sigma_h^\lambda} = \infty$, $j = 0$, and $k = \ell = 1$.

Step 3. Set $\ell = \ell + 1$. If $\ell = k$, then set $\ell = \ell + 1$. If $\ell > n$, then set $k = k + 1$ and $\ell = 1$. If $k > n$, set $k = \ell = 1$ and go to Step 4. Otherwise, construct sequence $\sigma_{k,\ell}$ from sequence σ_h^λ by interchanging jobs $\sigma_h^\lambda(k)$ and $\sigma_h^\lambda(\ell)$. If $\sigma_{k,\ell} \in S$, repeat Step 3; otherwise, set $S = S \cup \{\sigma_{k,\ell}\}$ and determine the worst-case absolute deviation from optimality for sequence $\sigma_{k,\ell}$ (denoted by $d_{\sigma_{k,\ell}}$) using Procedure Worst-Case [Discrete Scenarios]. If $d_{\sigma_{k,\ell}} < UB_{\sigma_h^\lambda}$, set $UB_{\sigma_h^\lambda} = d_{\sigma_{k,\ell}}$, $j = 1$, $k^* = k$, and $\ell^* = \ell$. Repeat Step 3.

Step 4. Set $\ell = \ell + 1$. If $\ell = k$, then set $\ell = \ell + 1$. If $\ell > n$, then set $k = k + 1$ and $\ell = 1$. If $k > n$, go to Step 5. Otherwise, construct sequence $\sigma'_{k,\ell}$ from sequence σ_h^λ by inserting job $\sigma_h^\lambda(k)$ into position ℓ. If $\sigma'_{k,\ell} \in S$, repeat Step 4; otherwise, set $S = S \cup \{\sigma_{k,\ell}\}$ and determine the worst-case absolute deviation from optimality for sequence $\sigma'_{k,\ell}$ using Procedure Worst-Case [Discrete Scenarios]. If $d_{\sigma'_{k,\ell}} < UB_{\sigma_h^\lambda}$ set $UB_{\sigma_h^\lambda} = d_{\sigma'_{k,\ell}}$, $j = 2$, $k^* = k$, and $\ell^* = \ell$. Repeat Step 4.

Step 5. If $UB_{\sigma_h^\lambda} \geq UB^\lambda$, go to Step 6; otherwise, set $UB^\lambda = UB_{\sigma_h^\lambda}$. If $j = 1$, set $\sigma_h^\lambda = \sigma_{k^*,\ell^*}$; else, set $\sigma_h^\lambda = \sigma'_{k^*,\ell^*}$. Set $UB_{\sigma_h^\lambda} = \infty$, $j = 0$, $k = \ell = 1$, and go to Step 3.

Step 6. If $UB^\lambda < UB$, set $UB = UB^\lambda$ and $\sigma_h = \sigma_h^\lambda$. If $\lambda = |\Lambda|$, stop; otherwise, go to Step 2.

4.2. *Processing time intervals*

A heuristic for the ADRS problem with processing time intervals has also been developed and tested. The approximation approach exploits a property that consistently yields good solutions. If we know the last critical job, and the sets L, R of jobs that occupy sequence positions prior and after the last critical job, respectively, then we can construct a sequence with promising performance by arranging the jobs $i \in L$ in Johnson order with respect to $\bar{p}_{i1}$ and $\underline{p}_{i2}$, followed by the last critical job, followed by jobs $i \in R$ arranged in Johnson order with respect to $\underline{p}_{i1}$ and $\bar{p}_{i2}$. This property is not necessarily characteristic of the optimal solution because the identity of the last critical job of any sequence constructed in this manner may change when the worst-case performance of this sequence is evaluated.

We employ an improvement approach that attempts to construct a good initial schedule in one pass. Since we know that each job i will contribute at least $\min\{\bar{p}_{i1}, \bar{p}_{i2}\}$ to the makespan of the robust schedule, we first compute $\min\{\bar{p}_{i1}, \bar{p}_{i2}\}$ for each job i, and assign $i \in L$ if $\bar{p}_{i1} \leq \bar{p}_{i2}$ and $i \in R$ if $\bar{p}_{i1} > \bar{p}_{i2}$. The rationale behind this strategy is that if $\bar{p}_{i1} \leq \bar{p}_{i2}$, then job i is more likely to provide its minimum contribution to the makespan of the approximate schedule if it is placed relatively early in the sequence. Given this assignment of jobs to L and R, a sequence is constructed by arranging jobs $i \in L$ in Johnson order with respect to $\bar{p}_{i1}$ and $\underline{p}_{i2}$, followed by jobs $i \in R$ arranged in Johnson order with respect to $\underline{p}_{i1}$ and $\bar{p}_{i2}$. The worst-case performance of this initial sequence is determined using Procedure Worst-Case [Processing Time Intervals]. We then seek to improve the initial solution by considering simple modifications to the heuristic sequence. As before, we generate the $n(n-1)$ insertions and $n(n-1)/2$ pairwise interchanges that can be derived from the incumbent sequence. Only those modifications that do not violate Propositions 3 and 4 are explicitly evaluated. The worst-case absolute deviation from optimality is then determined for each remaining sequence using Procedure Worst-Case [Processing Time Intervals]. If no improvement is realized over the initial heuristic solution, the procedure terminates. Otherwise, the alternative sequence yielding the minimum worst-case deviation from optimality becomes the incumbent sequence, and

the process is repeated. The process continues until the set of alternative sequences yields no improvement over the incumbent solution. The procedure is detailed below.

Procedure Improve [Processing Time Intervals]

Input. Processing time ranges, $[\underline{p}_{ij}, \bar{p}_{ij}]$, $i = 1, 2, \ldots, n$, $j = 1, 2$, for each job on each machine.

Output. Heuristic sequence, σ_h, that provides an Upper Bound (UB) on the minimum worst-case absolute deviation from optimality.

Step 1. Set $L = R = \emptyset$. For $i = 1, 2, \ldots, n$, if $\bar{p}_{i1} \leq \bar{p}_{i2}$, then $L = L \cup \{i\}$; otherwise, $R = R \cup \{i\}$. Construct sequence σ_h by arranging jobs in L in Johnson order with respect to machine 1 and 2 processing times of $\bar{p}_{i1}$ and $\underline{p}_{i2}$, respectively, followed by jobs in R arranged in Johnson order with respect to machine 1 and 2 processing times of $\underline{p}_{i1}$ and $\bar{p}_{i2}$, respectively. Determine the worst-case absolute deviation from optimality for sequence σ_h using Procedure Worst-Case [Processing Time Intervals], with UB set equal to d_{σ_h}. Set $S = \{\sigma_h\}$, $UB_{\sigma_h} = \infty$, $j = 0$, and $k = \ell = 1$.

Step 2. Set $\ell = \ell + 1$. If $\ell = k$, then set $\ell = \ell + 1$. If $\ell > n$, then set $k = k + 1$ and $\ell = 1$. If $k > n$, set $k = \ell = 1$ and go to Step 3. Otherwise, construct sequence $\sigma_{k,\ell}$ from sequence σ_h by interchanging jobs $\sigma_h(k)$ and $\sigma_h(\ell)$. If $\sigma_{k,\ell} \in S$, repeat Step 2; else, determine the worst-case absolute deviation from optimality for sequence $\sigma_{k,\ell}$ using Procedure Worst-Case [Processing Time Intervals]. If $d_{\sigma_{k,\ell}} < UB_{\sigma_h}$, set $UB_{\sigma_h} = d_{\sigma_{k,\ell}}$, $j = 1$, $k^* = k$, and $\ell^* = \ell$. Repeat Step 2.

Step 3. Set $\ell = \ell + 1$. If $\ell = k$, then set $\ell = \ell + 1$. If $\ell > n$, then set $k = k + 1$ and $\ell = 1$. If $k > n$, go to Step 4. Otherwise, construct sequence $\sigma'_{k,\ell}$ from sequence σ_h by inserting job $\sigma_h(k)$ into position ℓ. If $\sigma'_{k,\ell} \in S$, repeat Step 3; else, determine the worst-case absolute deviation from optimality for sequence $\sigma'_{k,\ell}$ using Procedure Worst-Case [Processing Time Intervals]. If $d_{\sigma'_{k,\ell}} < UB_{\sigma_h}$ set $UB_{\sigma_h} = d_{\sigma'_{k,\ell}}$, $j = 2$, $k^* = k$, and $\ell^* = \ell$. Repeat Step 3.

Step 4. If $UB_{\sigma_h} \geq UB$, stop; otherwise, set $UB = UB_{\sigma_h}$. If $j = 1$, set $S = S \cup \{\sigma_{k^*,\ell^*}\}$ and $\sigma_h = \sigma_{k^*,\ell^*}$; else, set $S = S \cup \{\sigma'_{k^*,\ell^*}\}$ and $\sigma_h = \sigma'_{k^*,\ell^*}$. Set $UB_{\sigma_h} = \infty$, $j = 0$, $k = \ell = 1$, and go to Step 2.

5. Computational results

To investigate the computational behavior of the solution approaches proposed for the robust scheduling problem, procedures were coded in FORTRAN and implemented on a IBM 486/66 MHz personal computer. The first set of experiments consisted of test problems involving n = 9, 12, and 15 jobs and $|\Lambda|$ = 4, 8, and 12 processing time scenarios. Processing times for each job $i = 1, 2, \ldots, n$ on each machine $j = 1, 2$ were randomly generated for each scenario $\lambda = 1, 2, \ldots, |\Lambda|$ from a uniform distribution of integers on the interval $[10\beta_j, (10 + 40\alpha)\beta_j]$, where α is a parameter that allows the variability of processing times across jobs in a given problem instance to be controlled, and β_j represents the relative processing requirement on machine j, thus allowing the location of processing bottlenecks to be controlled. Three values of α ($\alpha = 0.2, 0.6, 1.0$) and three vectors representing the relative processing requirements on machines 1 and 2 ($[\beta_1, \beta_2] = [1.0, 1.0]$, $[1.2, 1.0]$, $[1.0, 1.2]$) were included in the experimental design. Ten replications were generated for each combination of n, $|\Lambda|$, α, and $[\beta_1, \beta_2]$, resulting in a total of 810 test problems. Each problem instance was solved using the branch-and-bound algorithm and the improvement heuristic described in Section 3. In addition, the sequence that optimizes expected makespan performance was determined for each test problem using a branch-and-bound process that assumes that each processing time scenario is equally likely.

The results are shown in Table 1, which reports: (i) the number of sequences evaluated, the number of nodes in the associated solution tree, and CPU times (in seconds) for the branch-and-bound algorithm, (ii) the number of evaluated sequences and CPU times for the improvement heuristic; (iii) the average and maximum percentage increase over the optimal solution for approximations obtained from the improvement heuristic and from the schedule that optimizes expected makespan performance; and; (iv) the average and maximum percentage deviation between the expected makespan of the robust schedule the optimal expected makespan. Since the location of processing bottlenecks did not have an appreciable effect on computational performance or heuristic solution quality, the results in Table 1 are aggregated over the three values of $[\beta_1, \beta_2]$ included in the experimental design.

The results in Table 1 clearly illustrate the effect of problem size (n) on problem difficulty. However, the actual number of sequences evaluated by the branch-and-bound algorithm is quite small compared with the $n!$ possible sequences. Still, the algorithm demonstrated the capability to solve only small- to medium-sized problems, justifying the need for heuristic solution approaches. Optimal solutions were also increasingly difficult to obtain as the number of processing time scenarios was decreased, suggesting that the additional computational effort required to identify the worst-case scenario for each sequence is more than offset by the enhanced effectiveness of the lower bounding process as the number of processing time scenarios increases. The improvement heuristic required only modest computational effort to generate approximate solutions.

Table 1. Computational results – discrete processing time scenarios

			Branch-and-bound procedure			*Improvement heuristic*				*Optimal expected makespan schedule*		*Expected makespan of robust schedule*	
n	α	$\lvert\Lambda\rvert$	*Number of sequences*	*Tree size*	*CPU*	*Number of sequences*	*CPU*	*Mean*	*Max*	*Mean*	*Max*	*Mean*	*Max*
	0.2	4	40.4	561.2	0.5	976.0	0.3	0.0	0.0	12.3	63.2	0.2	0.9
	0.2	8	13.8	201.6	0.3	2004	0.5	0.0	0.0	18.9	95.8	0.3	1.3
	0.2	12	8.7	195.4	0.3	2999	0.8	0.0	0.0	23.8	134.2	0.5	2.1
	0.6	4	45.0	626.9	0.5	1197	0.3	0.0	0.0	15.0	91.7	0.1	0.6
9	0.6	8	21.9	379.1	0.4	2578	0.7	0.9	10.0	24.5	153.0	0.3	1.1
	0.6	12	12.2	245.0	0.4	4063	1.3	1.0	12.5	18.4	87.0	0.3	1.4
	1.0	4	38.4	671.7	0.5	1295	0.4	0.8	8.3	14.7	76.8	0.4	1.6
	1.0	8	20.5	266.5	0.4	2728	0.8	0.7	11.1	25.6	168.3	0.2	1.0
	1.0	12	15.8	205.1	0.3	4465	1.5	0.5	6.9	21.3	131.5	0.5	1.9
Avg [Max]			24.1	372.5	0.4	2478	0.7	0.4	[12.5]	19.4	[168.3]	0.3	[2.1]
	0.2	4	865.1	17 502	52.4	1922	0.5	0.0	0.0	19.1	80.4	0.3	1.5
	0.2	8	448.3	8 923	31.5	4109	1.1	0.0	0.0	18.3	92.3	0.3	1.2
	0.2	12	336.1	9 258	28.8	6312	1.9	0.0	0.0	25.2	121.6	0.6	2.7
	0.6	4	1 783	44 165	74.5	2520	0.8	0.0	0.0	17.2	69.5	0.2	0.9
12	0.6	8	989.0	16 238	57.5	5335	1.5	1.4	15.4	21.0	93.6	0.4	1.9
	0.6	12	226.5	4 306	16.7	9005	2.8	1.5	12.5	26.9	190.4	0.5	2.4
	1.0	4	5 124	102 437	160.2	2690	0.7	0.9	9.1	17.5	71.5	0.3	1.3
	1.0	8	1 009	16 723	61.8	6085	1.8	0.6	5.6	24.1	108.6	0.5	2.5
	1.0	12	71.8	2 194	9.5	10290	3.3	1.0	13.6	24.0	145.8	0.5	2.1
Avg [Max]			1 206	24 638	54.8	5363	1.6	0.6	[15.4]	21.5	[190.4]	0.4	[2.7]
	0.2	4	13 008	370 429	1231	3333	1.1	0.0	0.0	16.0	58.3	0.4	2.1
	0.2	8	5 597	162 818	672.4	7508	2.0	0.0	0.0	22.8	101.6	0.5	2.6
	0.2	12	2 896	115 132	561.6	11386	3.9	0.0	0.0	26.1	171.5	0.7	3.0
	0.6	4	16 321	1068 307	1937	4558	1.4	0.0	0.0	19.5	97.8	0.3	1.3
15	0.6	8	6 354	584 655	1134	9775	2.6	1.1	12.5	25.3	163.8	0.4	1.7
	0.6	12	3 126	216 693	816.0	15510	5.0	0.7	10.0	26.5	182.2	0.6	2.7
	1.0	4	28 155	3505 701	3016	4868	1.6	0.8	8.3	19.1	106.0	0.4	2.0
	1.0	8	15 448	1625 276	2022	11009	3.6	1.0	11.1	26.8	158.3	0.7	3.3
	1.0	12	6 821	343 487	984.5	17643	5.6	1.3	14.3	25.9	169.7	0.6	2.8
Avg [Max]			10 858	888 055	1375	9510	3.0	0.5	[14.3]	23.1	[182.2]	0.5	[3.3]

The results in Table 1 also indicate that the improvement heuristic consistently yielded close approximations of the robust schedule. The optimal solution was correctly identified in over 95% of the 810 test problems, while the error incurred in approximating the optimal solution averaged 0.5%, with a maximum of 15.4%. The solution quality associated with the schedule that optimizes expected makespan performance was considerably lower, with an average approximation error of 21.3%, and a maximum error of 190.4%. This observation indicates that optimizing average performance can and does yield schedules whose worst-case performance is quite poor. In contrast, the expected makespan performance of the robust schedule closely approximated the optimal expected makespan, with an average percentage deviation of 0.4%, and a maximum of 3.3%. This observation provides evidence that the robust schedule optimally hedges against the worst-case realization of job processing times while maintaining excellent expected makespan performance.

The second set of experiments again involves test problems with $n = 9, 12$, and 15 jobs. Test problems were generated by first randomly drawing the lower end of the processing time range for each job $i = 1, 2, \ldots, n$ on each machine $j = 1, 2$ from a uniform distribution of integers on the interval $\underline{p}_{ij} \in [10\beta_j, (10 + 40\alpha_1)\beta_j]$, where α_1 is a parameter that allows the variability of processing times across jobs in a given problem instance to be controlled, and β_j again represents the relative processing requirement on machine j. The upper end of the processing time range for each job i on machine j was next randomly drawn from a uniform distribution of integers on the interval $\bar{p}_{ij} \in [\underline{p}_{ij}, \underline{p}_{ij}(1 + \alpha_2)]$, where parameter α_2 controls the variability of a given job's processing time. Three values of α_1 ($\alpha_1 = 0.2, 0.6, 1.0$), three values of α_2 ($\alpha_2 = 0.2, 0.6, 1.0$), and three vectors representing the

relative processing requirements on machines 1 and 2 ($[\beta_1, \beta_2] = [1.0, 1.0], [1.2, 1.0], [1.0, 1.2]$) were included in the experimental design. Ten replications were generated for each combination of n, α_1, α_2, and $[\beta_1, \beta_2]$, resulting in a total of 810 test problems. Each problem instance was solved using the branch-and-bound algorithm and the heuristic discussed in Section 4. In addition, the sequence that optimizes expected makespan performance was directly determined by assuming independent, exponential processing time distributions fit to the minimum and maximum processing times specified for each job on each machine, and using the results of Cunningham and Dutta [12] (see also Pinedo [13] and Ku and Niu [14]). The results are presented in Table 2, which reports information (again aggregated over the three values of $[\beta_1, \beta_2]$) on computational performance and heuristic solution quality similar to that contained in Table 1.

As expected, the computational effort required to obtain optimal solutions using the branch-and-bound procedure grew rapidly with problem size, but again the number of evaluated sequences is small compared to $n!$, suggesting that the dominance results and bounding procedures were effective in simplifying the search for the robust schedule. Still, the branch-and-bound approach is clearly intractable for larger, practical problems. Computational effort also increased consistently with the parameter α_2, suggesting that problem difficulty is significantly affected by the variability of individual job processing times, as measured by the difference between $\underline{p}_{ij}$ and $\bar{p}_{ij}$. The computational requirements of the improvement heuristic remained quite low for all problem instances.

The results in Table 2 also indicate that the improvement heuristic consistently yielded close approximations of the robust schedule. The optimal solution was correctly identified in over 95% of the 810 test problems, while the error incurred in approximating the optimal solution averaged 0.2%, with a maximum of 16.7%. The

Table 2. Computational results – processing time intervals

			Branch-and-bound procedure			*Improvement heuristic*				*Optimal expected makespan schedule*		*Expected makespan of robust schedule*	
n	α_1	α_2	*Number of sequences*	*Tree size*	*CPU*	*Number of sequences*	*CPU*	*Mean*	*Max*	*Mean*	*Max*	*Mean*	*Max*
	0.2	0.2	1.0	17.6	0.1	104.3	0.1	0.0	0.0	0.0	0.0	0.1	0.8
	0.2	0.6	4.4	90.9	0.1	131.0	0.1	0.0	0.0	37.3	101.0	0.1	1.1
	0.2	1.0	4.1	252.8	0.2	157.7	0.1	0.9	12.5	24.7	51.6	0.5	2.6
	0.6	0.2	1.2	18.7	0.1	111.2	0.1	0.0	0.0	3.8	74.2	0.0	0.0
9	0.6	0.6	6.4	123.5	0.1	151.1	0.1	0.0	0.0	16.5	81.1	0.1	1.2
	0.6	1.0	7.1	164.6	0.1	157.3	0.1	0.4	11.1	18.4	93.8	0.4	3.7
	1.0	0.2	1.1	19.4	0.1	107.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.6	5.9	116.1	0.1	144.6	0.1	0.0	0.0	36.7	268.6	0.0	0.0
	1.0	1.0	28.1	758.9	0.3	154.5	0.1	0.0	0.0	31.9	192.5	0.3	3.9
Avg [Max]			6.6	173.6	0.1	135.5	0.1	0.1	[12.5]	18.7	[268.6]	0.2	[3.9]
	0.2	0.2	1.0	25.0	0.1	206.8	0.1	0.0	0.0	3.2	89.2	0.1	0.5
	0.2	0.6	13.7	368.5	3.2	275.3	0.2	0.0	0.0	62.1	228.5	0.1	1.0
	0.2	1.0	121.0	15 771	13.8	337.6	0.2	0.4	7.1	24.2	61.3	0.7	3.7
	0.6	0.2	1.4	23.7	0.1	212.9	0.1	0.0	0.0	0.0	0.0	0.0	0.0
12	0.6	0.6	39.6	858.4	6.5	225.4	0.1	0.0	0.0	11.4	53.9	0.1	1.5
	0.6	1.0	198.5	4 730	10.3	331.4	0.2	0.0	0.0	25.7	93.4	0.4	2.6
	1.0	0.2	1.0	33.4	0.1	206.7	0.1	0.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.6	15.0	676.5	4.1	244.1	0.2	0.6	16.7	32.9	187.0	0.2	0.2
	1.0	1.0	2134	132 425	116.4	312.7	0.2	0.3	5.3	31.6	133.3	0.1	0.8
Avg [Max]			280.6	17 212	17.2	261.4	0.2	0.1	[16.7]	21.2	[228.5]	0.2	[3.7]
	0.2	0.2	1.0	33.2	0.1	382.3	0.3	0.0	0.0	4.1	77.3	0.1	0.4
	0.2	0.6	818.3	206 343	273.9	492.6	0.4	0.6	16.7	55.8	261.2	0.1	1.8
	0.2	1.0	5351	930 696	1012	552.8	0.6	1.6	9.5	27.3	88.4	0.4	2.6
	0.6	0.2	16.0	543.7	0.5	332.1	0.3	0.0	0.0	0.0	0.0	0.0	0.0
15	0.6	0.6	851.3	69 740	91.2	442.5	0.4	0.3	10.0	28.3	109.5	0.1	0.5
	0.6	1.0	3399	239 661	275.8	512.7	0.5	0.2	6.3	57.1	241.3	0.2	1.8
	1.0	0.2	1.0	32.0	0.1	302.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0
	1.0	0.6	110.1	5 015	6.3	432.4	0.4	0.0	0.0	5.1	42.0	0.2	1.2
	1.0	1.0	596.7	92 385	136.8	442.5	0.4	0.2	6.5	58.6	289.1	0.0	0.0
Avg [Max]			1238	171 605	199.6	432.4	0.4	0.3	[16.7]	26.3	[289.1]	0.1	[2.6]

solution quality associated with the schedule that optimizes expected makespan performance was considerably lower, with an average approximation error of 22.1%, and a maximum error of 289.1%. In contrast, the expected makespan performance of the robust schedule closely approximated the optimal expected makespan, with an average percentage deviation of 0.2%, and a maximum of 3.9%. This observation suggests that the robust schedule provides an optimal hedge against the worst-case realization of job processing times without significant sacrifice with respect to expected makespan performance.

6. Conclusions

This paper has focused on two-machine flow shop environments in which the processing times of individual jobs are uncertain and the performance measure of interest is system makespan. We defined a measure of schedule robustness that considers exposure to the risk of poor system performance over all potential realizations of job processing times. We discussed two alternative frameworks for structuring processing time uncertainty, i.e., through a set of discrete processing time scenarios and through a set of independent processing time intervals. For the discrete scenario and processing time interval cases, the robust scheduling problem was formulated, and optimization and heuristic solution approaches were developed. Our computational results indicated that robust schedules hedge effectively against uncertain processing times while maintaining excellent expected makespan performance. High-quality approximations of the robust schedule were also consistently obtained using simple improvement heuristics.

Robust scheduling can be a powerful mechanism for managing the impact of manufacturing uncertainty on system performance. Further research should explore the relationship between sources of manufacturing variability, operational decisions affected by these uncertainties, and how the associated risk is incorporated into the decision-making process. Possible scheduling issues include the development of robust models for other scheduling environments where performance uncertainty varies with the schedule, investigation of other performance measures that are sensitive to manufacturing variability, and detailed consideration of other sources of and modeling frameworks for manufacturing uncertainty.

Acknowledgements

We thank the departmental editor and anonymous reviewers for their helpful comments and suggestions.

References

[1] Foley, R.D. and Suresh, S. (1984) Stochastically minimizing the makespan in flow shops. *Naval Research Logistics Quarterly*, **31**, 551–557.
[2] Pinedo, M. (1982) Minimizing the expected makespan in stochastic flow shops. *Operations Research*, **30**, 148–162.
[3] Möhring, R.H., Radermacher, F.J. and Weiss, G. (1984) Stochastic scheduling problems I: general strategies. *Zeitschrift fr Operations Research*, **28**, 193–260.
[4] Möhring, R.H., Radermacher, F.J. and Weiss, G. (1985) Stochastic scheduling problems II: set strategies. *Zeitschrift fr Operations Research*, **29**, 65–104.
[5] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G. and Shmoys, D.B. (1993) Sequencing and scheduling: algorithms and complexity, in *Handbooks in OR and MS*, Graves, S.C. *et al.* (eds.), vol 4, North Holland, Amsterdam. pp. 445–522.
[6] Righter, R. (1994) Stochastic scheduling, in *Stochastic Orders*, Shaked, M. and Shantikumar, G. (eds.), Academic Press, San Diego, CA.
[7] Daniels, R. and Kouvelis, P. (1995) Robust scheduling to hedge against processing time uncertainty in single stage production. *Management Science*, **41**, 363–376.
[8] Rosenblatt, J.J. and Lee, H.L. (1987) A robustness approach to facilities design. *International Journal of Production Research*, **25**, 479–486.
[9] Kouvelis, P., Kurawarwala A.A. and Gutierrez, G.J. (1992) Algorithms for robust single- and multiple-period layout planning for manufacturing systems. *European Journal of Operational Research*, **63**, 287–303.
[10] Mulvey, J.M., Vanderbei, R.J. and Zenios, S.A. (1995) Robust optimization of large-scale systems. *Operations Research*, **43**, 264–281.
[11] Johnson, S.M. (1954) Optimal two- and three-stage production schedules with set-up times included, *Naval Research Logistics Quarterly*, **1**, 61–68.
[12] Cunningham, A. and Dutta, S. (1973) Scheduling jobs with exponentially distributed processing times on two machines of a flow shop. *Naval Research Logistics Quarterly*, **16**, 69–81.
[13] Pinedo, M. (1984) Optimal policies in stochastic shop scheduling. *Annals of Operations Research*, **1**, 305–329.
[14] Ku, P.S. and Niu, S.C. (1986) On Johnson's two machine flow shop with random processing times. *Operations Research*, **34**, 130–136.
[15] Baker, K.R. (1994) Elements of sequencing and scheduling. Technical report, Dartmouth College, Hanover, NH.

Appendix

Proof of Theorem 1. We reduce the absolute deviation robust scheduling problem to the two-partition problem. Recall the two-partition problem: Given finite set I and size $s_i \in Z_+$ for $i \in I$, does there exist a subset $I_0 \subseteq I$ such that $\sum_{i\in I_0} s_i = \sum_{i\in I\setminus I_0} s_i$?

For a given finite set I and size $s_i \in Z_+$ for $i \in I$, construct an instance of the absolute deviation robust scheduling problem with two machines, two processing time scenarios, $n+1$ jobs (job j_0 and jobs $i \in I = \{1, 2, \ldots, n\}$). The following defines the processing time of job i on machine j in each of the two scenarios.

$$p_{ij}^1 = \begin{cases} (1/2)\sum_{i=1}^n s_i, & \text{for } i = j_0,\ j = 1,2, \\ 0, & \text{for } i \in I,\ j = 1, \\ s_i, & \text{for } i \in I,\ j = 2. \end{cases}$$

$$p_{ij}^2 = \begin{cases} (1/2)\sum_{i=1}^n s_i, & \text{for } i = j_0,\ j = 1,2, \\ 0, & \text{for } i \in I,\ j = 2, \\ s_i, & \text{for } i \in I,\ j = 1. \end{cases}$$

We claim that there exists a two-partition $\sum_{i\in I} s_i = \sum_{i\in I\setminus I_0} s_i$, with $I_0 \subset I$ if and only if the absolute robust scheduling problem has an optimal objective function value of zero. Observe that the Johnson sequence for both scenarios has a makespan of $(3/2)\sum_{i\in I} s_i$.

Assume that a set $I_0 \subseteq I$ of jobs are scheduled before job j_0, and the remaining $I\setminus I_0$ jobs are scheduled after j_0 in schedule σ_0. We examine three cases.

Case 1. $\sum_{i\in I_0} s_i > (1/2)\sum_{i\in I} s_i > \sum_{i\in I\setminus i_0} s_i$. For scenarios 1 and 2, we have:

$$\varphi(\sigma_0, \mathbf{P}^1) = \sum_{i\in I_0} s_i + \frac{1}{2}\sum_{i\in I} s_i + \sum_{i\in I\setminus I_0} s_i,$$

$$\varphi(\sigma_0, \mathbf{P}^2) = \sum_{i\in I_0} s_i + \frac{1}{2}\sum_{i\in I} s_i + \frac{1}{2}\sum_{i\in I} s_i.$$

Thus, $\max\{\varphi(\sigma_0, \mathbf{P}^1), \varphi(\sigma_0, \mathbf{P}^2)\} = \varphi(\sigma_0, \mathbf{P}^2) > \frac{3}{2}\sum_{i\in I} s_i$.

Case 2. $\sum_{i\in I_0} s_i < (1/2)\sum_{i\in I} s_i < \sum_{i\in I\setminus i_0} s_i$. For scenarios 1 and 2, we have:

$$\varphi(\sigma_0, \mathbf{P}^1) = \frac{1}{2}\sum_{i\in I} s_i + \frac{1}{2}\sum_{i\in I} s_i + \sum_{i\in I\setminus I_0} s_i,$$

$$\varphi(\sigma_0, \mathbf{P}^2) = \sum_{i\in I_0} s_i + \frac{1}{2}\sum_{i\in I} s_i + \sum_{i\in I\setminus I_0} s_i.$$

Thus, $\max\{\varphi(\sigma_0, \mathbf{P}^1), \varphi(\sigma_0, \mathbf{P}^2)\} = \varphi(\sigma_0, \mathbf{P}^1) > (3/2)\sum_{i\in I} s_i$.

Case 3. $\sum_{i\in I_0} s_i = (1/2)\sum_{i\in I} s_i = \sum_{i\in I\setminus I_0} s_i$.

Then $\varphi(\sigma_0, \mathbf{P}^1) = \varphi(\sigma_0, \mathbf{P}^2) = (3/2)\sum_{i\in I} s_i$ and $\max\{\varphi(\sigma_0, \mathbf{P}^1), \varphi(\sigma_0, \mathbf{P}^2)\} - (3/2)\sum_{i\in I} s_i = 0$. ■

Proof of Theorem 2. Consider sequence σ and processing time scenario λ, with $\sigma(i_0)$ the last critical job in the associated schedule. We examine four cases.

Case 1. $p_{i1} < \bar{p}_{i1}$ for some $i \in V_1^\sigma \cup \{\sigma(i_0)\}$. Increasing p_{i1} by an amount equal to ϵ results in an increase in the makespan of schedule σ, $\varphi(\sigma, \mathbf{P}^\lambda)$, of ϵ, while increasing the optimal system makespan, $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ by no more than ϵ (the increase in $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ equals ϵ if $i \in V_1^{\sigma_\lambda^*} \cup \{\sigma_\lambda^*(i_0)\}$, and is less than ϵ if $i \in V_2^{\sigma_\lambda^*}$). Thus, $\mathrm{d}(\sigma, \mathbf{P}^\lambda) = \varphi(\sigma, \mathbf{P}^\lambda) - \varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is nondecreasing in p_{i1}, and

$$\max_{\underline{p}_{i1} \le p_{i1} \le \bar{p}_{i1}} \mathrm{d}(\sigma, \mathbf{P}^\lambda)$$

is realized at $p_{i1} = \bar{p}_{i1}$.

Case 2. $p_{i2} < \bar{p}_{i2}$ for some $i \in V_2^\sigma \cup \{\sigma(i_0)\}$. Increasing p_{i2} by an amount equal to ϵ similarly results in an increase in $\varphi(\sigma, \mathbf{P}^\lambda)$ of ϵ, while increasing $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ by at most ϵ (the increase in $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ equals ϵ if $i \in V_2^{\sigma_\lambda^*} \cup \{\sigma_\lambda^*(i_0)\}$, and is less than ϵ if $i \in V_1^{\sigma_\lambda^*}$). Thus, $\mathrm{d}(\sigma, \mathbf{P}^\lambda) = \varphi(\sigma, \mathbf{P}^\lambda) - \varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is nondecreasing in p_{i2}, and

$$\max_{\underline{p}_{i2} \le p_{i2} \le \bar{p}_{i2}} \mathrm{d}(\sigma, \mathbf{P}^\lambda)$$

is realized at $p_{i2} = \bar{p}_{i2}$.

Case 3. $p_{i1} > \underline{p}_{i1}$ for some $i \in V_2^\sigma$. Decreasing p_{i1} by an amount equal to ϵ results in no change in $\varphi(\sigma, \mathbf{P}^\lambda)$, and does not increase $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ (the change in $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is 0 if $i \in V_2^{\sigma_\lambda^*}$, and is less than 0 if $i \in V_1^{\sigma_\lambda^*} \cup \{\sigma_\lambda^*(i_0)\}$). Thus, $\mathrm{d}(\sigma, \mathbf{P}^\lambda) = \varphi(\sigma, \mathbf{P}^\lambda) - \varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is nonincreasing in p_{i1}, and

$$\max_{\underline{p}_{i1} \le p_{i1} \le \bar{p}_{i1}} \mathrm{d}(\sigma, \mathbf{P}^\lambda)$$

is realized at $p_{i1} = \underline{p}_{i1}$.

Case 4. $p_{i2} > \underline{p}_{i2}$ for some $i \in V_1^\sigma$. Decreasing p_{i2} by an amount equal to ϵ results in no change in $\varphi(\sigma, \mathbf{P}^\lambda)$, and does not increase $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ (the change in $\varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is 0 if $i \in V_1^{\sigma_\lambda^*}$, and is less than 0 if $i \in V_2^{\sigma_\lambda^*} \cup \{\sigma_\lambda^*(i_0)\}$). Thus, $\mathrm{d}(\sigma, \mathbf{P}^\lambda) = \varphi(\sigma, \mathbf{P}^\lambda) - \varphi(\sigma_\lambda^*, \mathbf{P}^\lambda)$ is nonincreasing in p_{i2}, and

$$\max_{\underline{p}_{i2} \le p_{i2} \le \bar{p}_{i2}} \mathrm{d}(\sigma, \mathbf{P}^\lambda)$$

is realized at $p_{i2} = \underline{p}_{i2}$. ■

Proof of Proposition 1. Consider sequence σ in which job h precedes job i, and suppose that σ is the absolute deviation robust schedule. Construct sequence σ' from σ by interchanging the positions of jobs i and h, and assume that σ' is not a robust schedule. If λ_0 denotes the worst-case absolute deviation scenario for sequence σ, λ_0' the worst-case absolute deviation scenario for sequence σ', and $\lambda \in \Lambda$ any other processing time scenario, then:

$$\mathrm{d}(\sigma', \mathbf{P}^{\lambda_0'}) > \mathrm{d}(\sigma, \mathbf{P}^{\lambda_0}) \ge \mathrm{d}(\sigma, \mathbf{P}^\lambda).$$

Consider processing time scenario λ_0', and let C_{kj} and C_{kj}' denote the resulting completion time of job k on machine j in sequence σ and σ', respectively. Suppose jobs i and h occupy position H in sequences σ and σ', respectively. Then from Baker [15], the time required to complete the first H jobs is minimized by partitioning this set into subsets U (jobs k for which $p_{k1}^{\lambda_0'} \le p_{k2}^{\lambda_0'}$) and V (jobs k for which $p_{k1}^{\lambda_0'} > p_{k2}^{\lambda_0'}$) and constructing a schedule in which jobs in subset U are sequenced in nondecreasing order of

$p_{k1}^{\lambda_0'}$, followed by jobs in subset V sequenced in nonincreasing order of $p_{k2}^{\lambda_0'}$. Since $p_{i1}^{\lambda_0'} \leq p_{h1}^{\lambda_0'}$ and $p_{h2}^{\lambda_0'} \leq p_{i2}^{\lambda_0'}$ is guaranteed by the premise that $\bar{p}_{i1} \leq \underline{p}_{h1}$ and $\underline{p}_{i2} \geq \bar{p}_{h2}$, it follows that $C_{i2} \geq C'_{h2}$. Since the machine 2 completion time of job i in sequence σ is no less than that of job h in sequence σ', $\mathrm{d}(\sigma, \mathbf{P}^{\lambda_0'}) \geq \mathrm{d}(\sigma', \mathbf{P}^{\lambda_0'})$, resulting in a contradiction. Thus, sequence σ cannot be the absolute deviation robust schedule. ■

Proof of Proposition 2. Consider sequence σ in which job h immediately precedes job i, and suppose that σ is the absolute deviation robust schedule. Construct sequence σ' from σ by interchanging the positions of jobs i and h, and assume that σ' is not a robust schedule. If λ_0 denotes the worst-case absolute deviation scenario for sequence σ, λ_0' the worst-case absolute deviation scenario for sequence σ', and $\lambda \in \Lambda$ any other processing time scenario, then:

$$\mathrm{d}(\sigma', \mathbf{P}^{\lambda_0'}) > \mathrm{d}(\sigma, \mathbf{P}^{\lambda_0}) \geq \mathrm{d}(\sigma, \mathbf{P}^{\lambda}).$$

Suppose job ℓ immediately precedes jobs i and h in sequence σ and σ', respectively. Consider processing time scenario λ_0', and let C_{kj} and C'_{kj} denote the resulting completion time of job k on machine j in sequence σ and σ', respectively. Since $C_{i1} = C'_{h1} = C_{\ell 1} + p_{i1}^{\lambda_0'} + p_{h1}^{\lambda_0'}$, $C_{i2} = \max\{C_{i1}, C_{h2}\} + p_{i2}^{\lambda_0'}$, and $C_{h2} = \max\{C_{h1}, C_{\ell 2}\} + p_{h2}^{\lambda_0'}$, we obtain:

$$\begin{aligned}
C_{i2} &= \max\{C_{\ell 1} + p_{i1}^{\lambda_0'} + p_{h1}^{\lambda_0'}, C_{\ell 1} \\
&\quad + p_{h1}^{\lambda_0'} + p_{h2}^{\lambda_0'}, C_{\ell 2} + p_{h2}^{\lambda_0'}\} + p_{i2}^{\lambda_0'}, \\
C_{i2} &= \max\{C_{\ell 1} + p_{i1}^{\lambda_0'} + p_{h1}^{\lambda_0'} + p_{i2}^{\lambda_0'}, C_{\ell 1} \\
&\quad + p_{h1}^{\lambda_0'} + p_{h2}^{\lambda_0'} + p_{i2}^{\lambda_0'}, C_{\ell 2} + p_{h2}^{\lambda_0'} + p_{i2}^{\lambda_0'}\}, \\
C_{i2} &\geq \max\{C_{\ell 1} + p_{i1}^{\lambda_0'} + p_{h1}^{\lambda_0'} + p_{h2}^{\lambda_0'}, C_{\ell 1} \\
&\quad + p_{i1}^{\lambda_0'} + p_{i2}^{\lambda_0'} + p_{h2}^{\lambda_0'}, C_{\ell 2} + p_{h2}^{\lambda_0'} + p_{i2}^{\lambda_0'}\} = C'_{h2},
\end{aligned}$$

since either $p_{i1}^{\lambda_0'} \leq \min\{p_{h1}^{\lambda_0'}, p_{i2}^{\lambda_0'}\}$ or $p_{h2}^{\lambda_0'} \leq \min\{p_{h1}^{\lambda_0'}, p_{i2}^{\lambda_0'}\}$ is guaranteed by the premise that $\min\{\bar{p}_{i1}, \bar{p}_{h2}\} \leq \min\{\underline{p}_{h1}, \underline{p}_{i2}\}$. Since the machine 2 completion time of job i in sequence σ is no less than that of job h in sequence σ', $\mathrm{d}(\sigma, \mathbf{P}^{\lambda_0'}) \geq \mathrm{d}(\sigma', \mathbf{P}^{\lambda_0'})$, resulting in a contradiction. Thus, sequence σ cannot be the absolute deviation robust schedule. ■

Biographies

Panos Kouvelis is currently a Professor of Operations and Manufacturing Mangement at the Olin School of Business at Washington University in St. Louis. He received his Ph.D. in Industrial Engineering and Engineering Management at Stanford, M.B.A. and M.S. degree in Industrial and Systems Engineering at the University of Southern California, and a Diploma in Mechanical Engineering at the National Technical University of Athens. He has published over 40 articles in *Management Science*, *Operations Research*, *Naval Research Logistics*, *IIE Transactions* and other high quality academic journals. He published *Robust Discrete Optimization*, Kluwer in 1996 and *Global Operations and Logistics*, Wiley in 1998. He currently serves as department editor of *IIE Transactions* on Project Management and Associate Editor of *Management Science* on Supply Chain Management. His current research focuses on Operations Scheduling, Manufacturing Strategy, and Global Supply Chain Management.

Richard L. Daniels is Professor of Operations Management in the DuPree College of Management at the Georgia Institute of Technology. He received his Ph.D. in Operations Management from the Anderson Graduate School of Management at UCLA in 1986. His research interests include manufacturing planning and control systems, resource flexibility and its impact on operational efficiency, and decision making under uncertainty. His articles have appeared in a number of academic journals, including *Management Science*, *Operations Research*, *Naval Research Logistics*, and *IIE Transactions*.

George L. Vairaktarakis is the Lewis Progressive Assistant Professor in the Weatherhead School of Management at Case Western Reserve University. He received his Ph.D. in Industrial Engineering from the ISE Dept. at the University of Florida in 1994. Also, he holds M.Sc. degrees in Industrial Engineering and in Applied Mathematics. His research has been motivated by manufacturing companies in Milwaukee and the greater Cleveland area. His interests include Machine Scheduling, Workforce Planning, Project Management, International Operations Management, Combinatorial Optimization and Graph Theory. He has authored or co-authored more than 40 academic articles many of which have appeared in *Manufacturing & Service Operations Management*, *Operations Research*, *OR Letters*, *Naval Research Logistics*, *IIE Transactions* and the *Journal of Operations Management*. He is a member of INFORMS, the Decision Sciences Institute, and the American Society for Quality.

Contributed by the Scheduling Department