Probability in the Engineering and Informational Sciences, **21**, 2007, 579–595. Printed in the U.S.A.
DOI: 10.1017/S0269964807000332

STOCHASTIC BATCH SCHEDULING AND THE "SMALLEST VARIANCE FIRST" RULE

MICHAEL PINEDO
Stern School of Business
New York University
E-mail: mpinedo@stern.nyu.edu

Consider a single machine that can process multiple jobs in batch mode. We have n jobs and the processing time of job j is a random variable X_j with distribution F_j. Up to b jobs can be processed simultaneously by the machine. The jobs in a batch all have to start at the same time and the batch is completed when all jobs have finished their processing (i.e., at the maximum of the processing times of the jobs in that batch). We are interested in two objective functions, namely the minimization of the expected makespan and the minimization of the total expected completion time. We first show that under certain fairly general conditions, the minimization of the expected makespan is equivalent to specific deterministic combinatorial problems, namely the Weighted Matching problem and the Set Partitioning problem. We then consider the case when all jobs have the same mean processing time but different variances. We show that for certain special classes of processing time distributions the *Smallest Variance First* rule minimizes the expected makespan as well as the total expected completion time. In our conclusions we present various general rules that are suitable for the minimization of the expected makespan and the total expected completion time in batch scheduling.

1. INTRODUCTION

Consider a single machine and n jobs. The processing time of job j is a random variable X_j from distribution F_j. The machine can process jobs in a batch mode; that is, it can process up to b jobs at the same time; the b being the batch size. The jobs that are processed in any given batch start their processing all at the same time, and the completion time of the batch is determined by the last job of the batch to be completed. So the time to process a batch is equal to the maximum of the processing times of the jobs in the batch. Let C_j denote the completion time of job j. We are

 0269-9648/07 $25.00

interested in the policies that minimize the expected makespan $E(C_{\max}) = E(\max(C_1, \ldots, C_n))$ and policies that minimize the total expected completion time $E\left(\sum_{j=1}^{n} C_j\right)$.

There are many settings in industry where jobs have to be processed in batches; batch scheduling problems are actually ubiquitous. For example, burn-in operations in the production of circuit boards are performed in ovens that can handle several jobs at the same time. Other applications occur in the process industries. Chemical processes often have to be performed in tanks or kilns. In such cases too, a batching machine might have to process a number of jobs simultaneously, see Chandru et al. [2] and Hochbaum and Landy [3].

In the deterministic scheduling literature, batch scheduling has received a significant amount of attention over the years. This research has resulted in numerous articles as well as chapters in basic scheduling books. The problem considered in this article is the stochastic counterpart of a problem that is often referred to in the deterministic scheduling literature as the single-machine batch scheduling problem. The deterministic counterparts of our problems have at times been denoted by $\tilde{1} \mid b \mid C_{\max}$ and $\tilde{1} \mid b \mid \sum C_j$. These deterministic problems turn out to be polynomial time solvable via dynamic programming. The deterministic batch scheduling problem $\tilde{1} \mid b \mid C_{\max}$ is very easy. It is clear that an optimal schedule puts first the b longest jobs in a batch, then forms the second batch by taking among the remaining jobs the b longest jobs, and so on. The $\tilde{1} \mid b \mid \sum C_j$ problem is slightly more difficult. Brucker et al. [1] analyzed this problem and obtained an $O(n^{b(b-1)})$ dynamic programming algorithm.

Koole and Righter [4] were the first to consider stochastic counterparts of $\tilde{1} \mid b \mid C_{\max}$ and $\tilde{1} \mid b \mid \sum C_j$. They considered the minimization of $E(C_{\max})$ when the random processing times of the n jobs, $X_1, \ldots, X_n$, are stochastically ordered and showed that a simple full batch policy again minimizes the expected makespan in a class of arbitrary batch policies; that is, first group b jobs with the largest means; then group among the remaining jobs the b jobs with the largest means, and so on. (The last batch might turn out to have less than b jobs.) Koole and Righter [4] found that it is much harder to obtain elegant and comprehensive results for the minimization of the total expected completion time when the n processing times are stochastically ordered. Even when the n processing times are exponentially distributed with different means, no simple optimal policy could be found.

In this article we consider special cases in which all processing times have the same mean. We focus on how the variance (or the variability) of the distributions affects the schedule. In order to focus on the variance, we assume that the means of the processing times of all the jobs are the same, say 1. We are interested in determining the structure of the policies that minimize the expected makespan and the structure of the policies that minimize the total expected completion time.

This article is organized as follows. In the next section we discuss preliminaries with regard to processing time distributions, forms of stochastic dominance, and structures of optimal policies. In the third section we consider two special cases of the stochastic batching problem that are equivalent to two deterministic combinatorial problems, namely the Weighted Matching problem and the Set Partitioning problem. In the fourth section we determine the optimality of the *Smallest Variance First* (SVF)

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

policy when the processing time distributions have symmetric probability density functions and are ordered according to a form of stochastic dominance that we refer to as symmetrically more variable. In the subsequent section we consider distributions that are mixtures of exponential distributions and mixtures of Erlang distributions. We show that the SVF policy minimizes both the expected makespan and the total expected completion time in case the batch size b is 2. In the last section we present our conclusions and formulate heuristics and rules for the minimization of the expected makespan and the total expected completion time in more general settings.

2. PRELIMINARIES

In this article we consider two classes of scheduling policies. Neither class of policies allows for preemptions.

The first class of policies is referred to as the class of *full batch* policies. In this class, the decision-maker must, whenever he starts a batch of jobs, select a full batch provided there are b or more jobs still waiting for processing. In such a schedule, all batches are full, with the possible exception of the last batch, which might be a partial batch (since at the end of the process, there might be less than b jobs waiting for processing).

The second class of policies is referred to as the class of *arbitrary batch* policies. The decision-maker is allowed to start a partial batch (which contains strictly less than b jobs) at any time, even when b or more jobs are waiting for processing. It can be shown easily that in many settings it pays to have partial batches when the expected makespan or the total expected completion time has to be minimized.

In what follows we consider various classes of probability distributions. One class of distribution functions that we will consider is the class of symmetric distribution functions. A symmetric random variable X_j has a density function $f_j(t)$ that is symmetric around mean 1 and is defined on a finite support [0, 2] (i.e., $(f_j(t) = f_j(2 - t))$. Examples of such classes of distributions are as follows:

1. the Uniform distribution (either discrete or continous)
2. the truncated Normal distribution
3. the Beta distribution
4. the Binomial distribution.

The Uniform distribution with mean 1 on a support $[a, 2 - a]$ might be either continuous or discrete. A Normal distribution with mean 1 has to be truncated at zero as well as at 2 and has to be renormalized. Beta distributions have a finite support and can be symmetric around their mean. The Binomial distribution is symmetric when the probability parameter is 1/2.

Random variables with symmetric density functions have some nice properties. First, for the class of distribution functions defined above, it can be shown that the

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

variance is always less than or equal to 1. The distribution with the smallest (zero) variance is the deterministic distribution with mean 1; the distribution with the highest variance is the distribution that takes the value 0 with probability 0.5 and the value 2 with probability 0.5.

Assume $X_1, \ldots, X_b$ are symmetric random variables with mean 1 and defined over the support [0, 2]. From the fact that the random variable X_j has the same distribution as the random variable $2 - X_j$, it follows that $\max(X_1, \ldots, X_b)$ has the same distribution as $2 - \min(X_1, \ldots, X_b)$. So

$$E(\max(X_1, \ldots, X_b)) = 2 - E(\min(X_1, \ldots, X_b))$$

and

$$E(\min(X_1, \ldots, X_n)) = \int_0^2 \bar{F}_1(t) \times \bar{F}_2(t) \times \cdots \times \bar{F}_n(t)\, dt,$$

where $\bar{F}_j(t) = 1 - F_j(t)$.

A symmetric random variable X_1 with distribution F_1 on a support [0, 2] is said to be symmetrically more variable than a symmetric random variable X_2 with distribution F_2 on a support [0, 2] if $F_1(t) \geq F_2(t)$ for $0 \leq t \leq 1$ and $F_1(t) \leq F_2(t)$ for $1 \leq t \leq 2$. This form of stochastic dominance (which was used before by Pinedo [5]) will be denoted in what follows by $F_1 \geq_{\text{sv}} F_2$. Symmetric variability ordering is a more restricted form of stochastic dominance than the more widely used conventional variability ordering or convex ordering. It can be shown that if X_1 is symmetrically less variable than X_2, then X_1 is also less variable than X_2 according to the convex ordering. Since any symmetric distribution on [0, 2] is symmetrically less variable than the distribution that takes values 0 and 2 with probability 0.5, it is therefore also less variable according to the convex ordering. It follows that

$$\left(\frac{1}{2}\right)^{b-1} \leq E(\min(X_1, \ldots, X_b)) \leq 1$$

and

$$1 \leq E(\max(X_1, \ldots, X_b)) \leq 2 - \left(\frac{1}{2}\right)^{b-1}$$

when $X_1, \ldots, X_n$ are symmetrically distributed with mean 1.

3. EXPECTED MAKESPAN MINIMIZATION, WEIGHTED MATCHING, AND SET PARTITIONING

Often, it has turned out that stochastic scheduling problems are equivalent to certain (deterministic) combinatorial problems that at first sight might seem unrelated. This turns out to be the case here also. Some stochastic batch scheduling problems are, under certain conditions, equivalent to certain deterministic combinatorial problems.

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

We first consider the case in which $b = 2$ and the expected makespan has to be minimized. The processing time of job j is distributed according to an arbitrary distribution F_j.

LEMMA 1: *When $b = 2$, the policy that minimizes the expected makespan in the class of arbitrary batch policies is a full batch policy.*

PROOF: Suppose there are two or more batches with a single job. The original contribution of two batches with a single job to the expected makespan is $E(X_1) + E(X_2)$. Combine these two jobs and put them in a single batch. After putting them in a single batch their contribution is $E(\max(X_1, X_2))$ which is strictly less than $E(X_1) + E(X_2)$. ■

We now show that minimizing the expected makespan in this stochastic batch scheduling problem with $b = 2$, and arbitrary processing time distributions $F_1, \ldots, F_n$ is equivalent to a deterministic weighted matching problem, which can be solved in polynomial time. Assume the number of jobs n is even (if n would have been odd, then we would have added an additional job with zero processing time). Consider now a weighted matching problem with n nodes. Node j is connected with node k via an arc with a weight

$$w_{jk} = E(\max(X_j, X_k)) = E(X_j) + E(X_k) - E(\min(X_j, X_k)).$$

Clearly, the value w_{jk} can be determined. To find the pairs that have a minimum sum of total weights is equivalent to the deterministic weighted matching problem for which there exists a polynomial time algorithm (i.e., an algorithm that runs in $O(n^4)$. So, when $b = 2$, the problem can be solved in polynomial time even with *arbitrary* distributions $F_1, \ldots, F_n$, provided the expected minimum of two random variables can be computed easily. (Actually, Koole and Righter [4] found that when the n distributions are stochastically ordered, an even simpler algorithm minimizes the expected makespan.)

Consider now the case with b being an arbitrary fixed number larger than 2. It is to be expected that the problem is now significantly harder, since now the jobs have to be grouped in larger sets. We now have to make some additional assumptions with regard to the processing time distributions. Assume that each one of the random processing times has a mean of 1 and a symmetric probability density function over the support [0, 2]. The n symmetric distribution functions do not necessarily have to be of the same type; they just have to be symmetric. In what follows we show that the minimization of the expected makespan in the class of arbitrary policies can be formulated as a deterministic Set Partitioning problem. The Set Partitioning problem is a classical 0–1 integer program problem that is defined as follows:

$$\begin{aligned} &\min \quad \bar{c}\bar{x} \\ &\text{subject to} \\ &A\bar{x} = 1 \\ &x_j \in \{0, 1\}. \end{aligned}$$

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

The right-hand-side vector $\bar{1}$ is a vector of n 1's (the length of the vector being equal to the number of jobs in the scheduling problem). Each row in the constraint matrix corresponds to one of the jobs. The A matrix is a matrix of 0's and 1's. Each column in the constraint matrix corresponds to a set of jobs that can be put together in one batch. The maximum number of 1's in a column is b (the maximum size of a batch).

It remains to be shown how the $\bar{c}$ vector has to be set up. If the x variable corresponding to a given column (batch) is made equal to 1, then the corresponding component of the c vector must be equal to the expected time it takes to process that batch. If batch j consists of jobs $j_1, j_2, \ldots, j_k$, then the expected processing time of the batch is

$$E(\max(X_{j_1}, \ldots, X_{j_k})) = 2 - E(\min(X_{j_1}, \ldots, X_{j_k}))$$
$$= 2 - \int_0^2 (\bar{F}_{j_1}(t) \times \cdots \times \bar{F}_{j_k}(t))\, dt.$$

So

$$c_j = 2 - \int_0^2 (\bar{F}_{j_1}(t) \times \cdots \times \bar{F}_{j_k}(t))\, dt.$$

In general, it is well known that an arbitrary Set Partitioning problem is strongly NP-hard. In this case also, the number of columns in the A matrix is very large; that is,

$$\sum_{k=1}^{b} \frac{n!}{k!(n-k)!}.$$

However, we cannot draw the conclusion that this particular Set Partitioning problem is strongly NP-hard, since the $\bar{c}$ vector has a special structure with a fairly strong dependency between the different elements of the $\bar{c}$ vector.

In the next section we consider some more special cases of this problem that lead to more elegant optimal policies.

4. SYMMETRICALLY MORE VARIABLE DISTRIBUTIONS AND THE SMALLEST VARIANCE FIRST RULE

In this section we consider symmetric random variables with mean 1 that can be ordered according to the symmetric variability ordering. The random variable X_1 with distribution F_1 is said to be symmetrically more variable than the random variable X_2 with distribution F_2 if $F_1(t) \geq F_2(t)$ for $0 \leq t \leq 1$ and $F_1(t) \leq F_2(t)$ for $1 \leq t \leq 2$.

We assume that the n processing times $X_1, \ldots, X_n$ are ordered in such a way that $F_1 <_{\text{sv}} F_2 <_{\text{sv}} \cdots <_{\text{sv}} F_n$. We focus on both the expected makespan $E(C_{\max})$ and the total expected completion time $E\left(\sum C_j\right)$.

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

We first consider the case $b = 2$. In the previous section we have shown that when $b = 2$, a full batch policy minimizes the expected makespan in the class of arbitrary batch policies. We now show a similar result for the total expected completion time under the stricter distributional assumptions described earlier.

LEMMA 2: *If $b = 2$ and F_j, $j = 1, \ldots, n$, is a symmetric distribution function with mean 1, then a full batch policy minimizes the total expected completion time in the class of arbitrary batch policies.*

PROOF: Suppose there are two or more batches with a single job. First, we show that a single job cannot precede a batch of two jobs. An upper bound on the expected time it takes to process a batch of two jobs is 1.5. If a single job precedes a batch with two jobs, then these three jobs contribute a total of $1 + 2(1 + x) = 3 + 2x$, where x is the expected processing time of the batch with two jobs. If the batch of two jobs precedes the single job, then the contribution is $2x + x + 1 = 3x + 1$. If $x \leq 1.5$, then it is always better to have the batch with two jobs go first.

This implies that all batches consisting of a single job have to appear at the end of the schedule. Suppose now that we have two single-job batches following one another. Their contribution to the objective function is $1 + 2 = 3$. If they are put together in one batch, then the total expected completion time of these two jobs is at most 3, since the upper bound for a batch of two jobs is 1.5. ■

Actually, the result stated in Lemma 2 can be generalized. The lemma also holds for arbitrary distributions $F_1, \ldots, F_n$ with mean 1 on support [0, 2] (the distributions do not necessarily have to be symmetric for Lemma 2 to hold). However, in the remaining part of this section, the symmetry assumption on the distribution functions is crucial.

We are now ready for our main result when $b = 2$ and $F_1 \leq_{\text{sv}} \cdots \leq_{\text{sv}} F_n$.

THEOREM 1: *If $b = 2$ and $F_1 \leq_{\text{sv}} \cdots \leq_{\text{sv}} F_n$, then the full batch SVF rule minimizes both $E(C_{\max})$ and $E\left(\sum C_j\right)$ in the class of arbitrary batch policies.*

PROOF: Consider a schedule with two consecutive batches with two jobs each. The two batches consist of jobs j_1, j_2, j_3, and j_4. Assume $F_{j_1} \leq_{\text{sv}} F_{j_2} \leq_{\text{sv}} F_{j_3} \leq_{\text{sv}} F_{j_4}$. Assume the first batch consists of jobs j_1 and j_3 and the second batch consists of jobs j_2 and j_4. Let this schedule be denoted by π'. The two batches contribute the amount

$$\begin{aligned} A'_{\pi} &= E(\max(X_{j_1}, X_{j_3})) + E(\max(X_{j_2}, X_{j_4})) \\ &= 4 - E(\min(X_{j_1}, X_{j_3})) - E(\min(X_{j_2}, X_{j_4})) \\ &= 4 - \int_0^2 \bar{F}_{j_1}(t)\bar{F}_{j_3}(t)\,dt - \int_0^2 \bar{F}_{j_2}(t)\bar{F}_{j_4}(t)\,dt \end{aligned}$$

to the expected makespan. It can be shown easily that the expected processing time of the first batch is less than the expected processing time of the second batch. Consider an

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

interchange between jobs j_2 and j_3. We have to show that the new contribution to the expected makespan, $A_{\pi''}$, is lower than the old contribution $A_{\pi'}$:

$$
\begin{aligned}
A_{\pi''} - A_{\pi'} &= \int_0^2 \bar{F}_{j_1}(t)\bar{F}_{j_3}(t)\,dt + \int_0^2 \bar{F}_{j_2}(t)\bar{F}_{j_4}(t)\,dt \\
&\quad - \int_0^2 \bar{F}_{j_1}(t)\bar{F}_{j_2}(t)\,dt - \int_0^2 \bar{F}_{j_3}(t)\bar{F}_{j_4}(t)\,dt \\
&= \int_0^2 \bar{F}_{j_1}(t)\Big(\bar{F}_{j_3}(t) - \bar{F}_{j_2}(t)\Big)dt \\
&\quad - \int_0^2 \bar{F}_{j_4}(t)\Big(\bar{F}_{j_3}(t) - \bar{F}_{j_2}(t)\Big)dt \\
&= \int_0^2 \Big(\bar{F}_{j_1}(t) - \bar{F}_{j_4}(t)\Big)\Big(\bar{F}_{j_3}(t) - \bar{F}_{j_2}(t)\Big)dt \\
&= \int_0^1 \Big(\bar{F}_{j_1}(t) - \bar{F}_{j_4}(t)\Big)\Big(\bar{F}_{j_3}(t) - \bar{F}_{j_2}(t)\Big)dt \\
&\quad + \int_1^2 \Big(\bar{F}_{j_1}(t) - \bar{F}_{j_4}(t)\Big)\Big(\bar{F}_{j_3}(t) - \bar{F}_{j_2}(t)\Big)\,dt.
\end{aligned}
$$

It is easy to see that both integrands in the last expression are negative. So the contribution of the two batches to the expected makespan is reduced by pairing the two jobs with the smaller variances with one another and by pairing the two jobs with the larger variances with one another. So the expected makespan is reduced by the interchange.

Moreover, the expected processing time of the first batch after the interchange (with jobs j_1 and j_2) is less than the expected processing time of the first batch before the interchange (with jobs j_1 and j_3). This implies that the total expected completion time is also reduced by the interchange.

Suppose that job j_1 was originally paired with job j_4 and job j_2 was paired with j_3. One of these two batches will have a smaller expected processing time than the other (but we might not know which one). Assume that the batch with the smaller expected processing time goes first in the original schedule. Doing a similar pairwise interchange as was done earlier again reduces the sum of the expected processing times of the two batches as well as the expected processing time of the batch with the smaller expected processing time. This implies that the makespan as well as the total expected completion time is reduced by the interchange. This completes the proof of the theorem. ■

Note that the proof of the theorem does not only show that the SVF rule minimizes the expected makespan and the total expected completion time; it actually shows that SVF minimizes the expected time of the kth batch completion for $k = 1, \ldots, \lceil n/2 \rceil$.

However, the SVF rule minimizes the completion time of the kth batch in expectation, but not *stochastically*. In order to see why this minimization is not in the stochastic sense, consider the completion time of the very first batch and the probability of the

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

first batch being completed at time 0. One can easily concoct examples where the Largest Variance First rule has a higher probability of having the first batch completed by time 0. Thus, the SVF rule does not minimize the first batch completion stochastically.

In any case, the result in Theorem 1 can be generalized in the following direction: Assume job j has a weight w_j and the objective to be minimized is the sum of the expected weighted completion times $\left(\text{i.e.}, E\left(\sum w_j C_j\right)\right)$. If the weights and the distribution functions are "agreeable" in such a way that

$$w_1 \geq w_2 \geq \cdots \geq w_n$$

and

$$F_1 \leq_{\text{sv}} F_2 \leq_{\text{sv}} \cdots \leq_{\text{sv}} F_n,$$

then it can be easily shown that the SVF rule minimizes $E\left(\sum w_j C_j\right)$ in the class of full batch policies. Note that the SVF rule in the weighted case does not necessarily minimize the objective in the class of arbitrary batch policies. (In the class of arbitrary batch policies, it might be advantageous to put just a single job in one batch, provided its weight is sufficiently high.)

In the remaining part of this section we consider the case where $b \geq 3$. It can be shown easily that when $b \geq 3$, a partial batch policy might minimize either the expected makespan or the total expected completion time.

We consider now the class of full batch policies with $b \geq 3$. Consider two full batches that have to be processed consecutively on the machine. Both the first batch and the second batch have b jobs. Assume that the jobs in the first batch are jobs $j_1, \ldots, j_b$ and the jobs in the second batch are jobs $j_{b+1}, j_{b+2}, \ldots, j_{2b}$. The total time to produce these two batches is

$$\begin{aligned} &E(\max(X_{j_1}, \ldots, X_{j_b})) + E(\max(X_{j_{b+1}}, \ldots, X_{j_{2b}})) \\ &\quad = 4 - E(\min(X_{j_1}, \ldots, X_{j_b})) - E(\min(X_{j_{b+1}}, \ldots, X_{j_{2b}})). \end{aligned}$$

Assume that after these two batches have been completed, the total number of jobs remaining to be processed is m. The processing of these two batches contribute the amount

$$(m + 2b)(2 - E(\min(X_{j_1}, \ldots, X_{j_b}))) + (m + b)(2 - E(\min(X_{j_{b+1}}, \ldots, X_{j_{2b}})))$$

to the total completion time of all jobs. In the next theorem we assume that the $2b$ jobs are ordered according to symmetric variability.

THEOREM 2: *If two full batches follow one another in a schedule, then the schedule that puts the b jobs with the smaller variances in the first batch and the b jobs with the larger variances in the second batch minimizes the expected makespan as well as the total expected completion time.*

PROOF: Consider two full batches that appear one after another consecutively. The two batches contain together $2b$ jobs. Assume that the expected processing time of the first batch is less than the expected processing time of the second batch. Let S^l

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

denote the set of b jobs with the lowest variances among these $2b$ jobs and let S^h denote the set of b jobs with the highest variances. In the original partition of the jobs among the two batches, the first batch contains a subset of S^l, say set S_1^l, and a subset of set S^h, say subset S_1^h. The second batch contains a subset of S^l, say set S_2^l, and a subset of set S^h, say subset S_2^h. Clearly, subsets S_1^l and S_1^h together contain b jobs and subsets S_2^l and S_2^h together contain b jobs.

We now perform an interchange between subsets S_2^l and S_1^h. It is clear that the job with the smallest variance in subset S_1^h has a higher variance than the job with the largest variance in subset S_2^l; in what follows, we assume that each one of these two subsets contain k jobs. The job with the smallest variance in subset S_2^h has a larger variance than the job with the largest variance in subset S_1^l; both of these subsets contain $b - k$ jobs.

Let π' denote the original schedule and π'' denote the schedule after the interchange; let $A_{\pi'}$ and $A_{\pi''}$ denote the contribution to the makespan by schedules π' and π'', respectively.

Thus,

$$
\begin{aligned}
A_{\pi''} - A_{\pi'} &= \int_0^2 \prod_{j\in S_1^l} \bar{F}_j(t) \prod_{j\in S_1^h} \bar{F}_j(t)\,dt + \int_0^2 \prod_{j\in S_2^l} \bar{F}_j(t) \prod_{j\in S_2^h} \bar{F}_j(t)\,dt \\
&\quad - \int_0^2 \prod_{j\in S_1^l} \bar{F}_j(t) \prod_{j\in S_2^l} \bar{F}_j(t)\,dt - \int_0^2 \prod_{j\in S_1^h} \bar{F}_j(t) \prod_{j\in S_2^h} \bar{F}_j(t)\,dt \\
&= \int_0^2 \prod_{j\in S_1^l} \bar{F}_j(t) \left(\prod_{j\in S_1^h} \bar{F}_j(t) - \prod_{j\in S_2^l} \bar{F}_j(t) \right) dt \\
&\quad - \int_0^2 \prod_{j\in S_2^h} \bar{F}_j(t) \left(\prod_{j\in S_1^h} \bar{F}_j(t) - \prod_{j\in S_2^l} \bar{F}_j(t) \right) dt \\
&= \int_0^2 \left(\prod_{j\in S_1^l} \bar{F}_j(t) - \prod_{j\in S_2^h} \bar{F}_j(t) \right) \left(\prod_{j\in S_1^h} \bar{F}_j(t) - \prod_{j\in S_2^l} \bar{F}_j(t) \right) dt \\
&= \int_0^1 \left(\prod_{j\in S_1^l} \bar{F}_j(t) - \prod_{j\in S_2^h} \bar{F}_j(t) \right) \left(\prod_{j\in S_1^h} \bar{F}_j(t) - \prod_{j\in S_2^l} \bar{F}_j(t) \right) dt \\
&\quad + \int_1^2 \left(\prod_{j\in S_1^l} \bar{F}_j(t) - \prod_{j\in S_2^h} \bar{F}_j(t) \right) \left(\prod_{j\in S_1^h} \bar{F}_j(t) - \prod_{j\in S_2^l} \bar{F}_j(t) \right) dt.
\end{aligned}
$$

It can be shown easily that the integrands in the last two integrals are negative for all t. Thus, the sum of the expected processing time of the two batches is reduced by the interchange, implying that the interchange reduces the expected makespan.

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

The fact that the expected processing time of the first one of the two batches is also reduced by the interchange implies that the total expected completion time of all jobs is reduced by the interchange as well. ■

Theorem 2 has as an immediate consequence that if the number of jobs n is a multiple of the batch size b, then the SVF rule minimizes the expected makespan as well as the total expected completion time in the class of full batch policies. Actually, the SVF rule then minimizes the expected completion time of the kth batch for each k. If the number of jobs is one more than a multiple of the batch size, then the SVF rule also minimizes the expected makespan as well as the total expected completion time in the class of full batch policies (it can be shown easily that the job with the largest variance should be processed on its own at the very end of the process and all other jobs have to be processed according to SVF).

When n is neither a multiple of b nor 1 plus a multiple of b, then the problem is significantly harder. In a full batch policy, the last batch has to be a partial batch with more than one job. However, it might now not necessarily be the case that the jobs with the larger variances have to be grouped together in this last batch. One can easily concoct examples in which jobs with smaller variances have to be grouped together in the last batch (which is a partial batch).

Example: Consider the case with $b = 4$ and $n = 6$. Two jobs have deterministic processing times equal to 1. The remaining four jobs are stochastic with the same distribution: The processing time is zero with probability 0.5 and 2 with probability 0.5. The scheduling policy that minimizes the total expected completion time in the class of full batch policies as well as in the class of arbitrary batch policies is the full batch Largest Variance First policy; that is, the four stochastic jobs are processed first in a single batch, followed by the two deterministic jobs in a second batch. The expected total completion time in this case is 53/4. No other arbitrary batch policy is optimal.

5. MIXTURES OF EXPONENTIALS AND ERLANGS AND THE SMALLEST VARIANCE FIRST RULE

In the previous section we considered symmetric distributions on a finite support. The variances of the processing time distributions in the last section have a fairly tight upper bound (i.e., $\mathrm{Var}(X_j) \leq 1$ for all j). In this section we consider distributions on an infinite support (i.e., $[0, \infty]$) and the processing times can have larger variances. We consider two classes of distributions.

The first class of distributions, which in what follows is referred to as Class I, is a mixture of three exponentials. The random variable X_j is defined as follows:

1. exponential with mean 0 with probability p_j
2. exponential with mean 1 with probability $1 - 2p_j$
3. exponential with mean 2 with probability p_j.

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

So the rates of the three exponential distributions are ∞, 1, and 1/2, respectively. More formally, this class of distributions can be defined as

$$F_j(x) = 1 - (1 - 2p_j)e^{-x} - p_j e^{-x/2}.$$

Clearly, the mean of a distribution from this class is always 1 (i.e., $E(X_j) = 1$). However, the variance of X_j depends on the mixing probability p_j, and $\mathrm{Var}(X_j) = 1 + 4p_j$.

The second class of distributions, which is in what follows referred to as Class II, is a mixture of three Erlang distributions. The random variable X_j is now defined as follows:

1. exponential with mean 0 with probability p_j
2. exponential with mean 1 with probability $1 - 2p_j$
3. a convolution of two exponentials with means 1 with probability p_j.

Formally, this distribution can be defined as

$$F_j(x) = 1 - (1 - p_j)\, e^{-x} - p_j\, x e^{-x}.$$

Again, the mean of the distribution is 1, but now $\mathrm{Var}(X_j) = 1 + 2p_j$.

We have shown already in Lemma 1 that partial batches are not optimal when $b = 2$ and the expected makespan has to be minimized; that is, if there are two or more jobs waiting for processing, then two jobs have to be combined in a batch and have to start their processing together. At the end of the schedule there might be a batch with a single job.

Suppose we combine two jobs with processing times X_j and X_k in one batch. The expected time to process this batch is

$$E(\max(X_j, X_k)) = \Big(E(X_k) + E(X_j) + E(D_{jk})\Big)/2,$$

where

$$E(D_{jk}) = E(\max(X_j, X_k) - \min(X_j, X_k))$$

If there are n jobs, then a total of $\lceil n/2 \rceil$ batches have to be scheduled. If n is even, the n jobs have to be paired with one another in $n/2$ batches. If n is odd, then the last batch will consist of a single job.

If the jobs are scheduled in the order 1,2, . . . , n and n is even, then

$$E(C_{\max}) = E(\max(X_1, X_2)) + E(\max(X_3, X_4)) + \cdots + E(\max(X_{n-1}, X_n));$$

if n is odd, then

$$E(C_{\max}) = E(\max(X_1, X_2)) + E(\max(X_3, X_4)) + \cdots + E(\max(X_{n-2}, X_{n1})) + E(X_n).$$

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

Since

$$E(\max(X_j, X_k)) = \Big(E(X_k) + E(X_j) + E(D_{jk})\Big)/2,$$

it turns out that when n is even minimizing the expected makespan is equivalent to minimizing

$$E(D_{12}) + E(D_{34}) + \cdots + E(D_{n-1,n}),$$

and when n is odd, it is equivalent to

$$E(D_{12}) + E(D_{34}) + \cdots + E(D_{n-2,n-1}).$$

THEOREM 3: *The full batch SVF rule minimizes $E(C_{\max})$ in the class of arbitrary batch policies when $b = 2$ and the n processing time distributions are either all of Class I or all of Class II.*

PROOF: We first consider the case in which all n distributions are of Class I. It can be shown through straightforward analysis that if X_j and X_k are distributed according to Class I distributions, then

$$E(D_{jk}) = 1 + \frac{2}{3}p_j + \frac{2}{3}p_k - \frac{2}{3}p_j p_k.$$

In the formulas to be minimized, the $E(D_{jk})$ can now be replaced by $p_j + p_k - p_j p_k$. We have to make a distinction between two subcases, namely the subcase with n being even and the subcase with n being odd. When n is even, the objective to be minimized is

$$(p_1 + p_2 - p_1p_2) + (p_3 + p_4 - p_3p_4) + \cdots + (p_n + p_{n-1} - p_{n-1}p_n),$$

which is equivalent to maximizing

$$p_1p_2 + p_3p_4 + \cdots + p_{n-1}p_n.$$

It is clear that in order to maximize this last expression, the job with the smallest p_j has to be combined with the job that has the second smallest, and so on. So the job with the smallest variance has to be paired with the job with the second smallest variance, the job with the third smallest variance has to be paired with the job with the fourth smallest variance, and so on. This implies that the SVF policy is optimal. Clearly, SVF is not the only optimal policy. Actually, in order to minimize the expected makespan, only the pairing of the jobs is important. After the jobs have been paired in batches, it does not matter in which sequence the batches are processed.

When n is odd, the expression to optimize is slightly different, because now there will be one batch (the last one) that has a single job. So the expression to be

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

minimized is

$$(p_1 + p_2 - p_1p_2) + (p_3 + p_4 - p_3p_4) + \cdots + (p_{n-2} + p_{n-1} - p_{n-2}\,p_{n-1}).$$

First, we show that the job with the largest variance must be the one that goes alone in the last batch. Suppose that the job with the largest variance has been paired with another job in a batch and that another job with a smaller variance has been scheduled last as the job in the batch. An interchange between these two jobs reduces the expected makespan. This implies that the set of $n - 1$ jobs that do not include the job with the largest variance have to be paired with one another in $(n - 1)/2$ batches. The problem now reduces again to the problem of maximizing

$$p_1p_2 + p_3p_4 + \cdots + p_{n-2}p_{n-1}.$$

So again, the SVF rule minimizes the expected makespan.

We now consider the case in which all n distributions are of Class II; that is, they are distributed according to a mixture of Erlang distributions:

$$F_j(x) = 1 - (1 - p_j)e^{-x} - p_jxe^{-x}.$$

It can be shown through straightforward analysis that if X_j and X_k are distributed according to a Class II distribution, then

$$E(D_{jk}) = 1 + \frac{1}{2}p_j + \frac{1}{2}p_k - \frac{1}{2}p_jp_k.$$

Even though the expression for $E(D_{jk})$ is slightly different for Class II distributions than they are for Class I distributions, the arguments that have to be used in order to prove that the SVF policy is optimal are exactly the same. ■

Consider now the minimization of the total expected completion time. It turns out that if $b = 2$, a full batch policy might not necessarily be optimal in the class of arbitrary batch policies when the total expected completion time has to be minimized and the processing time distributions are either all of Class I or all of Class II (this is in contrast to the result in Lemma 2). A counterexample can be found easily: Consider the case with $b = 2$ and $n = 2$. Suppose that the two jobs are both of Class I and $p_1 = p_2 = 1/2$. It can be shown easily that the total expected completion time is less when the two jobs are processed one after another than processed together in a single batch of two jobs. The same is true when the two jobs are both of Class II with $p_1 = p_2 = 1/2$.

In the remaining part of this section we analyze scheduling policies that minimize the total expected completion time in the class of full batch policies.

When n is even, the total expected completion time is

$$E\left(\sum_{j=1}^{n} C_j\right) = nE(\max(X_1, X_2)) + (n - 2)E(\max(X_3, X_4))$$

$$+ \cdots + 2E(\max(X_{n-1}, X_n)),$$

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

and when n is odd, the total expected completion time is

$$E\left(\sum_{j=1}^{n} C_j\right) = nE(\max(X_1, X_2)) + (n-2)E(\max(X_3, X_4)) + \cdots + 3E(\max(X_{n-2}, X_{n-1})) + E(X_n).$$

Minimizing the expected total completion time when n is even is equivalent to minimizing

$$nE(D_{12}) + (n-2)E(D_{34}) + \cdots + 2E(D_{n-1,n}),$$

and when n is odd, it is equivalent to minimizing

$$nE(D_{12}) + (n-2)E(D_{34}) + \cdots + 3E(D_{n-2,n-1}).$$

THEOREM 4: *If $b = 2$ and the n processing times are either all of Class I or all of Class II, then the full batch SVF policy minimizes $E\left(\sum C_j\right)$ in the class of full batch policies.*

PROOF: Let $E(B_{j_i})$ denote the expected completion time of the ith batch in the sequence. We have shown already that the SVF rule minimizes the expected completion time of the last batch (which is equal to the expected makespan). This last batch is the $\lceil n/2 \rceil$th batch to be completed. In order to minimize the expected completion time of the batch immediately preceding the last batch, we proceed as follows. In the previous theorem we showed that the expected makespan increases linearly in each p_j. That means that if we would like to minimize the expected completion time of the batch immediately preceding the last one, then we should assign the one or two jobs (dependent on whether n is odd or even) with the largest variances to the last batch and keep all remaining jobs with smaller variances for the $\lceil n/2 \rceil - 1$ preceding batches. Applying Theorem 3 to these $2(\lceil n/2 \rceil - 1)$ jobs again specifies that these jobs have to be scheduled according to the SVF rule. Proceeding in this manner we can show that the expected time of the kth batch completion ($k = 1, \ldots, \lceil n/2 \rceil$) is minimized in expectation by the SVF rule. (However, the kth batch completion is not minimized stochastically.) ■

The result of Theorem 4 can be generalized to the weighted case as well (in a way similar to the weighted case considered in Section 4). If the weights are agreeable with the processing time distributions in such a way that $w_1 \geq w_2 \geq \cdots \geq w_n$ and $p_1 \leq p_2 \leq \cdots \leq p_n$, then the full batch SVF policy minimizes the total expected weighted completion time in the class of full batch policies.

6. CONCLUDING REMARKS

The SVF rule is not a very common rule in stochastic scheduling. SVF has turned out to be useful in the scheduling of stochastic flow shops (see Pinedo and Wie [15]), but it

https://doi.org/10.1017/S0269964807000332 Published online by Cambridge University Press

has never been considered useful in the scheduling of parallel machines. In the scheduling of parallel machines it has been shown in the past that the Largest Variance First rule minimizes several different objectives; see Pinedo and Weiss [6].

In the batch scheduling problem considered in this article, it turns out that the SVF rule minimizes the expected makespan and the total expected completion time at the same time. It does not occur often in scheduling theory that the same rule minimizes the expected makespan as well as the total expected completion time.

The results obtained in this article suggest the following two rules of thumb for minimizing the expected makespan and the total expected completion time, respectively. In order to minimize the expected makespan, one would tend to use a minimum number of batches. The jobs to be combined in a batch have to be as similar as possible; that is, jobs with large means and large variances have to be kept together and jobs with small means and small variances also have to be kept together. The order in which the batches are processed clearly does not affect the expected makespan. In order to minimize the total expected completion time, one might not always want to have a minimum number of batches. The order in which the batches are processed is, of course, very important for the minimization of the total expected completion time. The early batches in the schedule have to contain jobs with small means and small variances and the later batches have to contain jobs with large means and large variances.

If the variances in the processing times tend to be small, then it is more likely that a full batch policy should be adopted, whereas in the case of large variances, it is more likely that a partial batch policy should be adopted.

One important special case has not been considered in this article: the case with the batch size b being ∞. Minimizing the expected makespan is then trival: All the jobs have to be combined in a single batch. However, minimizing the total expected completion time is much harder. The deterministic counterpart of this problem has a nice structure, since it can be shown that the optimal schedule has to take the form of a shortest processing time first batch (SPT-batch) schedule; that is, if the jobs are indexed such that the processing times satisfy $p_1 \leq p_2 \leq \cdots \leq p_n$, then the schedule has to be such that adjacent jobs have to be grouped in batches. However, the number of jobs grouped in a batch might vary from one batch to the next. For example, a schedule might be of the form $(\{1, 2, 3\}, \{4\}, \{5, 6, 7, 8\}, \{9, 10\})$. It is clear that, at times, a batch should not be used to its fullest capacity. Consider the stochastic version of this problem in which the processing times of the jobs are stochastic with mean 1 and symmetrically distributed. It is easy to find examples in which one does not combine all jobs within a single batch. However, one still might ask the following question: If the jobs are indexed in increasing order of their variances, is the optimal schedule of the form of an SVF batch schedule? That is, do adjacent jobs have to be grouped in batches?

It might be of interest in the future to study generalizations of the models described in this article. One can, for example, study generalizations in which the jobs are subject to precedence constraints; that is, a given job can only be started when a certain set of other jobs have already been completed. Another generalization

could assume that the jobs are subject to compatibility constraints (or incompatibility constraints); that is, a job can only be put together in the same batch with another job if they are "compatible."

References

1. Brucker, P., Gladky, A., Hoogeveen, J.A., Kovalyov, M.Y., Potts, C.N., Tautenhahn, T., & Van De Velde, S.L. (1998). Scheduling a batching machine. *Journal of Scheduling* 1: 31–54.
2. Chandru, V., Lee, C.-Y., & Uzsoy, R. (1993). Minimizing total completion time on a batch processing machine with job families. *Operations Research Letters* 13: 61–65.
3. Hochbaum, D.S. & Landy, D. (1997). Scheduling semiconductor burn-in operations to minimize total flow time. *Operations Research* 45: 874–885.
4. Koole, G. & Righter, R. (2001). A stochastic batching and scheduling problem. *Probability in the Engineering and Informational Sciences* 15: 465–479.
5. Pinedo, M. (1982). Minimizing the expected makespan in stochastic flow shops. *Operations Research* 30: 148–162.
6. Pinedo, M. & Weiss, G. (1987). The "largest variance first" policy in some stochastic scheduling problems. *Operations Research* 35: 884–891.
7. Pinedo, M. & Wie, S.-H. (1986). Inequalities for stochastic flow shops and job shops. *Stochastic Models and Data Analysis* 2: 61–69.
8. Potts, C.N. & Kovalyov, M.Y. (2000). Scheduling and batching: A review. *European Journal of Operational Research* 120: 228–249.